

exemplary values of industrial equipment, health care equipment, security systems, etc. A step 303a could also comprise initiator 102' obtaining physical location 304a for device 101, which could comprise a room number, zone in a building or floor, shelf number, rack number, etc. Physical location 304a could also comprise a set of geographical coordinates for a value of physical location 304a or (ii) estimated geographical coordinates or similar physical location of device 101. In exemplary embodiments, initiator 102' can comprise a mobile phone or smart phone and can have a GPS or similar receiver for outdoor location, or use WiFi or Bluetooth for approximate indoor locations, and record the location in a physical location 304a.

Although step 303a to collect an identity list 304 for device 101 is depicted as after secure session setup 302a (confirming mutual authentication between initiator 102' and DPP server 103 in session 302a), step 303a to collect an identity list 304 could be conducted at other times, such as before Figure 3a or after a initiator user 102u takes initiator 102 to the approximate physical location for installation or configuration of device 101, including possibly during a step 202 above in Figure 2a. Or, in other exemplary embodiments the use of an identity list 304 can optionally be omitted, if initiator 102' and DPP server 103 can record the set of credentials 109 for a selected network access point 105, where the selected network access point 105 can be recorded without an identity list 304 (such as if the correct or preferred network access point 105 is known to DPP server 103 before a read step 202 in Figure 2a). As depicted in Figure 3a, initiator 102' can then send DPP server 103 a message 305 through secure session 302a, where message 305 can include ID-token.device 206 (depicted and described for message 204 in Figure 2a above) and the identity list 304. DPP server 103 can receive message 305 and record the data received in a server database 103a. The values of Physical Location 304a and Monitored Unit ID 304b can be recorded for support of subsequent operation of device 101 after a configuration step 106, and DPP server 103 could also send the values to another server supporting device 101 during operation such as a reporting server (not shown). DPP Server 103 can then send ID-token.device 206 through secure session 302a to device database 104 in a message 306.

A step 302d for DPP server 103 can also comprise DPP server 103 querying a server database 103a for an initiator bootstrap keys sequence number 197 for device 101 using ID-token.device 206 (or ID.device 101i). In exemplary embodiments, a device 101 with responder 101x could use multiple different initiator bootstrap public keys Bi 104a over time, such as updating the initiator bootstrap public key Bi 104a using the steps depicted and described in connection with Figure 6a below after a DPP 191 and configuration step 106 has been previously completed. The multiple possible reasons and benefits for updating and initiator bootstrap public keys Bi 104a are discussed with a step 608 in Figure 6a below. A server database 103a could record a value initiator bootstrap keys sequence number 197 in order to track which initiator bootstrap public key Bi 104a is currently recorded in responder 101x for device 101. For other embodiments, a responder 101x in

device 101 could also record a plurality of different initiator bootstrap public keys Bi 104a concurrently, such as a first initiator bootstrap public key Bi 104a for a first network 107 and a second initiator bootstrap public key Bi 104a for a second network 107. Responder 101x could use the multiple different, concurrently available initiator bootstrap public keys Bi 104a to concurrently
5 listen for a DPP authentication request message 122 from different possible networks 107. In a step 302d, A DPP server 103 could use an initiator bootstrap keys sequence number 197 to select the current, preferred initiator bootstrap public key Bi 104a for conducting a DPP 191 by an initiator 102'. DPP server 103 can send a database 104 a message 307 with an identity for device 101 of ID-token.device 206 (or identity ID.device 101i), and the selected initiator bootstrap keys sequence
10 number 197.

At step 306, DPP server 103 can select the mode of operation for initiator 102' to conduct a DPP 191 with device 101. The selected mode of operation for initiator 102' can comprise a mode 306a or an initiator mode 306a, which could be recorded in a server database 103a as depicted and described in connection with Figure 3b below. The selected mode 306a can specify the distribution
15 of PKI keys between DPP server 103 and initiator 102'. The selected mode 306a can determine the subsequent series of steps and message flows between DPP server 103 and initiator 102'. A first mode 306a could (a) specify that DPP server 103 records all public and private bootstrap and ephemerals keys for initiator 102', plus the public keys Br 101a and Pr 101e for responder 101x such that (b) resulting cryptographic operations and ECDH key exchanges and symmetric ciphering reside
20 within DPP server 103 and results are transmitted to initiator 102'. This first mode 306a could represent the message flows and steps in Figure 3a. A second mode 306a could specify that (i.a) initiator bootstrap PKI keys 104a and 104b reside within DPP server 103 and (i.b) responder bootstrap and ephemeral public keys for responder 101x reside within initiator 102', such that (ii) resulting cryptographic operations and key exchanges are shared between DPP server 103 and
25 initiator 102'. This second mode 306a could represent the message flows and steps depicted in Figure 5a.

A third mode 306a selected in a step 306 for DPP server 103 could specify the same recording of bootstrap and ephemeral keys as the second mode, with the addition that DPP server 103 receives and uses the initiator ephemeral private key pi 102b, also with resulting cryptographic
30 operations and key exchanges are shared between DPP server 103 and initiator 102' (but with a different allocation than the second mode). This third mode 306a could represent the message flows and steps depicted in Figure 5b. In exemplary embodiments, all of the first, second, and third modes operate with initiator bootstrap private key bi 104b residing within DPP server 103 and not shared with initiator 102'. By not sharing initiator bootstrap private key bi 104b with initiator 102' (which
35 could be a device with much less security than DPP server 103), the initiator bootstrap private key bi 104b can remain secured within network 107. These embodiments where initiator bootstrap private

key bi 104b remains in network 107 (e.g. in DPP server 103 and device database 104) support securely sharing the same initiator bootstrap public key Bi 104a potentially across a plurality of device 101 with responders 101x. Sharing the same initiator bootstrap public key Bi 104a across multiple devices supports recording the key Bi 104a in a device 101 during manufacturing. This simplifies and enables the use of a device provisioning protocol 191 across multiple devices 101 with mutual authentication, where the initiator 102' does not need to receive or be provisioned with the initiator bootstrap private key bi 104b.

In other words, without the technology described in the present invention, a device provisioning protocol with mutual authentication can essentially shift the problem of (a) securely mutually authenticating a device and providing credentials 109 to a device 101 to a new, but related problem of (b) securely providing an initiator bootstrap private key bi 104b to an initiator 102* . The first, second, and third modes 306a can solve this shifted/new problem since the initiator bootstrap private key bi 104b can permanently reside in a network 107 and does not need to be sent to the initiator 102' or an initiator 102* . However, with the technology in the present invention, the initiator 102' can still transmit and receive the full set of a device provisioning protocol 191 messages with a responder 101x as specified in DPP specification version 1.0.

A fourth mode 306a could specify that initiator 102' records all of the initiator bootstrap and ephemeral PKI keys, as well as the responder 101x public keys, such that resulting cryptographic operations and key exchanges reside in initiator 102' . This fourth mode 306a could represent the message flows and steps depicted in Figure 1b or Figure 1c, and in this case initiator 102' could comprise an initiator 102* . Other possibilities exist as well without departing from the scope of the present invention. Additional details for the different modes 306a selected in a step 306, with the resulting different allocation or recording of PKI keys are described below in Figure 3b for a server database 103a.

At step 308a, DPP server 103 can process data received in message 304 and the received identity list 304. DPP server 103 can use the values in a networks available list 304c in order to select an access network 112 for device 101. A step 308a could comprise DPP server 103 selecting a primary access network 308aa and a backup access network 308ab for a device 101, as depicted and described in connection with Figure 3b below. The selected access networks 308aa and 308ab for device 101 in a step 308a could be based on criteria such as bandwidth cost, expected energy or power requirements (i.e. 3G/4G may use more power than LPWAN), signal strength measurements recorded with a networks available list 304c, and commercial terms such as agreements between a (i) device owner for device 101 and (ii) networks in a networks available list 304c. DPP server 103 could also query other servers not depicted in Figure 3a to select an access network 112 from the networks available list 304c. In other words, a preferred access network 112 in the form of a primary access network 308aa for device 101 may need to be selected by DPP server 103 in a step

308a before the credentials 109 for the access network (or network access point 105) can be obtained. Although Figure 3a depicts DPP server 103 as conducting a step 308a in order to select a primary access network 308aa and backup access network 308ab, in exemplary embodiments a different server than DPP server 103 in a network 107 could (i) securely receive the networks
5 available list 304c and (ii) select the primary access network 308aa and backup access network 308ab.

After the selection of a primary access network 112 (308aa) and a backup access network 112 (308ab) in a step 308a, DPP server 103 could perform a step 308b depicted in order to obtain a set of network credentials 109 for both the selected primary access network 308aa and the selected
10 backup access network 308ab. For embodiments where (a) a network 107 operating DPP server 103 comprises a selected access network 308ab or 308ab, then (b) the credentials 109 for device 101 can be obtained from another server in a network 107 from Figure 1a and recorded in DPP server 103. For example, if a network access point 105 for network 107 is selected as an access network 112 for device 101, then network 107 would normally already record the set of credentials 109 for access
15 point 105. For embodiments where a selected access network 112 (e.g. selected primary access network 308aa) for device 101 other than network 107 operates the access point 105, then for a step 308b DPP server 103 could query the access network 112 for the equivalent of credentials 109. Additional details regarding embodiments where a different network than network 107 provide credentials for device 101 are depicted and described in connection with Figure 3b and Figure 6b
20 below.

Note that in exemplary embodiments, a DPP server 103 can preferably conduct a step 308a to select an access network 112 and a step 308b to obtain credentials 109 for the selected access network 112 before continuing with a device provisioning protocol 191. The reason is that if either
25 (i) a network cannot be selected potential from a networks available list 304c or (ii) credentials 109 cannot be obtained for the selected network, then there may be no reason to continue with a device provisioning protocol 191 since DPP server 103 may not have credentials 109 to send to device 101. In Figure 3a, although a step 308a and step 308b are depicted as taking place after DPP server 103 sends message 307 with a query for PKI keys to device database 104, in some embodiments a step 308a and step 308b could take place before DPP server 103 sends message 307 to device database
30 104.

Further, in some exemplary embodiments where initiator 102' uses network access point 105 with SSID.network-AP 109a with a preshared secret key (PSK) or similar authentication (as an access network 112 for initiator 102'), then a message 305 could comprise initiator 102' sending DPP server 103 the set of network credentials 109 used by initiator 102'. DPP server 103 could then
35 (i) record the credentials 109 in a server database 103a further described below in Figure 3b, (ii) use the value SSID.network-AP 109a as a selected primary access network 308aa, and (ii) use the

received credentials 109 from initiator 102' in a message 305 as the credentials 109 for device 101. Other possibilities regarding the source of credentials 109 exist without departing from the scope of the present invention.

Device database 104 can receive message 307 with ID-token.device 206 (or similarly ID.device 101i) and initiator bootstrap keys sequence number 197. Device database in step 309 can conduct a query of device database 104 for data associated with ID-token.device 206 and initiator bootstrap keys sequence number 197. In exemplary embodiments, a query in a step 309 can select an initiator bootstrap private key bi 104b and an initiator bootstrap public key Bi 104a using the initiator bootstrap keys sequence number 197 from a device table 104d for device 101. In this manner, for embodiments where initiator bootstrap PKI keys for device 101 are updated over time (e.g. such as depicted and described in connection with Figure 6a), then the most recently updated initiator bootstrap PKI keys can be selected by a database query in a step 309. The query in step 309 can return values for a message 310 from device database 104 to DPP server 103. The values in a message 310 can comprise data for ID-token.device 206 and the values can include responder bootstrap public key Br 101a, initiator bootstrap public key Bi 104a, initiator bootstrap private key bi 104b, a device identity ID.device 101i, a selected set of cryptographic parameters 199a for the PKI keys, and optionally PKEX key 198. Although not depicted in Figure 3a, message 310 could also include ID-token.device 206 so that DPP server 103 can track which of a possible plurality of initiators 102' and devices 101 that message 310 is associated with. In some embodiments, a step 302d and the use of a sequence number 197 can be omitted, and a query in step 309 and message 310 can return the most recently updated or recorded values for responder bootstrap public key Br 101a, initiator bootstrap public key Bi 104a, and initiator bootstrap private key bi 104b.

For embodiments where (A) a DPP session 191 between initiator 102' and device 101 fails due to a possible mismatch between (i) initiator bootstrap PKI keys selected in a query for step 309 and (ii) the actual initiator bootstrap public key Bi 104a recorded and used by a responder 101x in device 101 (such as the case where device 101 has undergone a "factory reset" to restore previously recorded initiator bootstrap public key Bi 104a), then (B) DPP server 103 could send a second message 307 with a request to use the oldest recorded initiator bootstrap PKI keys from a device table 104d (such as possibly the lowest or first recorded sequence number 197 for device 101 in device table 104d). For these embodiments where (A) a DPP session 191 between initiator 102' and device 101 fails due to a possible mismatch between (i) initiator bootstrap PKI keys selected in a query for step 309 and (ii) the actual initiator bootstrap public key Bi 104a recorded and used by a responder 101x in device 101, then (B) a query in step 309 can be for the oldest recorded initiator bootstrap PKI keys from a device table 104d. In other exemplary embodiments, DPP server 103 could send message 307 with a specific initiator bootstrap keys sequence number 197, and a query in a step 309 could be for a specific set of responder bootstrap public key Br 101a, initiator bootstrap

public key Bi 104a, initiator bootstrap private key bi 104b designated by the initiator bootstrap keys sequence number 197 in a device table 104d. Further, a key table 104k in a device database 104 can record the most recently updated values and keys for a device 101, while a device table 104d can record the historical values and PKI keys for a device 101 used over time, where the historical different values or PKI keys can be tracked, queried, or selected using an initiator bootstrap keys sequence number 197.

DPP server 103 can receive the message 310 and record the data received in a server database 103a. As depicted in Figure 3a, DPP server 103 can then send message 311 to initiator 102', where message 311 can include (i) the selected mode 306a of operation for initiator 102' and DPP server 103 for conducting a DPP 191 with responder 101x, (ii) a selected set of cryptographic parameters 199a for use with PKI keys and a device provisioning protocol, and (iii) optionally a PKEX key 198 for conducting a public key exchange with responder 101x. Message 311 can also optionally include device identity ID-token.device 206 in order to associate the data in message 311 with a particular responder 101x, since initiator 102' could potentially communicate with a plurality of different devices 101 over time.

At step 303b initiator 102' can receive and process the message 311. Initiator 102' and DPP server 103 can begin operating in the selected mode 306a for conducting a device provisioning protocol 191. Initiator 102' and DPP server 103 can apply and begin using the selected set of cryptographic parameters 199a. For a step 308b, if a PKEX key 198 is present in a message 311, then initiator 102' can conduct a public key exchange protocol with responder 101x in order to obtain responder bootstrap public key Br 101a via WiFi radio 102w and "in band". The public key exchange protocol is specified in the DPP specification version 1.0 section 5.6. If PKEX is optionally used between responder 101x and initiator 102', then initiator 102' could receive the responder bootstrap public key Br 101a and send the key Br 101a to DPP server 103 (and in this case DPP server 103 or database 104 may not have previously recorded responder bootstrap public key Br 101a). The use of a PKEX protocol and key 198 can also be optionally omitted in some embodiments, if responder bootstrap public key Br 101a can be recorded by a network 107 prior to a step 303b.

As depicted in Figure 3a, the collection of steps and messages before a step 313 can be grouped together into a step 301. Step 301 with the collective group of sub-steps and messages depicted for a step 301 can also be performed in other embodiments depicted in Figure 5a and Figure 5b below. Subsequent messages and steps depicted in Figure 3a starting after step 301 can represent a first mode 306a for DPP server 103 and initiator 102', where the allocation of recording PKI keys and symmetric keys from key exchanges for the first mode 306a are depicted below in Figure 3b.

A step 313 can comprise deriving an initiator ephemeral private key pi 102b and an initiator ephemeral public key Pi 102a, and in Figure 3a the DPP server 103 is depicted as conducting the

step 313 to derive the initiator ephemeral PKI keys. Step 313 can use a random number generator to produce a random number, which could be input into a key pair generation algorithm using the set of cryptographic parameters 199a. The resulting PKI key values from conducting a step 313 can comprise the initiator ephemeral private key pi 102b and the initiator ephemeral public key Pi 102a.

5 The resulting PKI key values from conducting a step 313, if conducted by a DPP server 103, can be recorded in a server database 103a, as depicted in Figure 3b below. Other embodiments depicted below contemplate a step 313 to derive initiator ephemeral PKI keys being conducted by an initiator 102'. In summary, some steps depicted as conducted by a DPP server 103 in Figure 3a could be conducted instead by an initiator 102' for other embodiments such as those depicted in Figure 5a and

10 Figure 5b, with a corresponding change in supporting message flows between DPP server 103 and initiator 102'.

Step 314 can comprise DPP server 103 conducting a first initiator key exchange step 314 using an Elliptic Curve Diffie Hellman (ECDH) key exchange 401a, as depicted and described in connection with Figure 4a below. Data input for a step 314 can comprise the initiator ephemeral private key pi 102b and the recorded responder bootstrap public key Br 101a. Data output from a

15 step 314 can comprise a key $k1$ as described in Figure 4a below. Step 314 can comprise the key exchange and symmetric key derivation step in DPP specification version 1.0 section 6.2.2 in order to for an initiator to obtain key $k1$.

At step 315, DPP server 103 can generate a random number for an initiator nonce 317a using parameters 199a. As used herein and throughout the present invention, the words "random number" can also mean a pseudo random number, since random numbers generated by nodes may not be truly random. However, their "randomness" may be sufficient in order to securely conduct the cryptographic operations contemplated herein. DPP server 103 can also determine the initiator capabilities, which could comprise a configurator for the initiator as depicted in Figure 3a (where responder 101x would be the enrollee). In other embodiments, the initiator 102' could have initiator capabilities as an enrollee (where the responder 101x could be the configurator). A step 315 can

20 comprise encrypting the initiator nonce 317a and initiator capabilities using the derived key $k1$ from step 314 in order to create a ciphertext 315a. An encryption step 315 step is further depicted and described in connection with Figure 4a below.

At step 316, DPP server 103 can calculate secure hash values for initiator bootstrap public key Bi 104a and responder bootstrap public key Br 101a, using a secure hash algorithm 199e. The values are depicted in Figure 3a as " $H(Bi)$ " and " $H(Br)$ ", respectively. DPP server 103 can then send initiator 102' message 317, which can include the hash value for the initiator bootstrap public key Bi 104a and the responder bootstrap public key Br 101a, the derived initiator ephemeral public key Pi 102a, and ciphertext 315a, where ciphertext 315a is encrypted with key $k1$ and includes an

30

35

initiator nonce 317a and the initiator capabilities. Initiator 102' can receive message 317 via the secure session 302b.

For a step 316a (not depicted in Figure 3a, but conducted by initiator 102' after receipt of message 317 and also depicted in Figure 5a below), initiator 102' can then send DPP Authentication Request message 122 to responder 101x, using WiFi radio 102w with initiator configuration 130. In exemplary embodiments, initiator 102' sends message 122 as a unicast message to responder 101x using MAC.R 101d as the destination/recipient, and initiator 102' received MAC.R 101d in a previous message such as message 220. Alternatively, initiator 102' sends message 122 as a broadcast message (and MAC.R 101d would not normally be required for broadcast). However, unicast for WiFi can be preferred, since unicast messages are acknowledged at the data link layer. In a step 316a, initiator 102' may need to repeat message 122 multiple times on several different channels in a channel list 130b from initiator configuration 130, since (i) responder 101x may be periodically sleeping in order to conserve battery power, and (ii) responder 101x may also be cycling through a channel list 130b in order to listen for message 122 on different channels (and the cycling by initiator 102' and responder 101x may not be synchronized).

In addition, DPP authentication request message 122 in Figure 3a can have some differences than message 122 in Figure 1b and Figure 1c. Message 122 in Figure 3a can originate from an initiator proxy 102' (where initiator proxy 102' may not record an initiator bootstrap private key bi 104b for responder 101x), while message 122 in Figure 1c can be from an initiator 102*, where initiator 102* can record and operate on the initiator bootstrap private key bi 104b. In other words, from the perspective of a responder 101x message 122 in Figure 1c and Figure 3a could be identical, but the initiator 102* and 102' can be different, depending on the recording and processing of PKI keys and shared secret keys. Further, although a message 122, message 123, etc. as depicted herein shows key values of "k1", "ke", etc., the depiction of the keys is to illustrate that ciphertext is encrypted with the corresponding key by the use of brackets. In other words, although the key values are shown with the messages, the actual key values are not transmitted inside or with the message in exemplary embodiments. The equivalent syntax is used in DPP specification version 1.0.

Responder 101x can receive message 122 and perform a series of steps to process the message and respond to the initiator 102'. Responder 101x in device 101 can compare (i) hash values received in message 122 with (ii) the calculated hash values for bootstrap public keys from a step 302c. If (a) the hash values match or are equal, then responder 101x can proceed to subsequent steps, and otherwise (b) continue to cycle through a channel list 130b in order to listen for incoming messages 122. At step 318, responder 101x can also generate two random numbers using a random number generator. The first random number can be used as a responder ephemeral private key pr 101f. The responder ephemeral private key pr 101f can be input into an ECC key pair generation

algorithm in order to output a responder ephemeral public key Pr 101e. The set of cryptographic parameters 199a used for the responder ephemeral PKI key pair can match the recorded parameters 199a used with the bootstrap public keys such as responder bootstrap public key Br 101a. The second random number generated by responder 101x can be used as a responder nonce 318b.

5 Responder 101x can also determine responder capabilities 319b, which could comprise the capabilities for an enrollee.

Responder 101x can conduct a first responder key exchange step 319, where the details for a key exchange step 319 are depicted in Figure 4a below. In summary, the received initiator ephemeral public key Pi 102a (from message 122) and the responder bootstrap private key br 101b can be input

10 into an ECDH key exchange in order to mutually derive a shared secret key M 410. The shared secret key M 410 can be input into a key derivation function 402a in order for responder 101x to derive mutually shared key $k1$ 314a. A responder 101x can then conduct a decryption step 320 as depicted in Figure 4a below. Responder can input the received ciphertext 315a into a symmetric ciphering algorithm 405b and key $k1$ 314a using parameter 199f in order to read the plaintext 403,

15 which could comprise the initiator nonce 317a and the initiator capabilities. In exemplary embodiments, the initiator 102' can have capabilities of a configurator, and responder 101x can select responder capabilities or an enrollee in a step 320. Steps 319 and steps 320 can also comprise the first four paragraphs of section 6.2.3 of DPP specification version 1.0.

Responder 101x can then conduct a second responder key exchange step 314b, where the

20 details for a key exchange step 314b are depicted and described in connection with Figure 4b below. In summary, the initiator ephemeral public key Pi 102a received in message 122 can be input into an ECDH key exchange algorithm 401a along with the responder ephemeral private key pr 101f derived in a step 318 above in order to output a secret shared key N 411. The value for shared secret key N 411 can be recorded and used in the derivation of a key ke 327a in a third key exchange step 321

25 below. For a step 314b, the value for the shared secret key N 411 can be input into a key derivation function 402b in order to derive a mutually shared symmetric key $k2$ 314c.

Responder 101x can conduct a third responder key exchange step 321, where the details for a key exchange step 321 are depicted in Figure 4c below. In summary, both (i) the derived responder ephemeral private key pr 101f from a step 318 with the recorded responder bootstrap private key br 101b, and (ii) the recorded initiator bootstrap public key Bi 104a can be input into an ECDH key

30 exchange algorithm 401c in order to output a secret shared key L 412. For embodiments where a DPP responder authentication 141 is conducted by responder 101x without the use of an initiator bootstrap public key Bi 104a, then calculation of a secret shared key L 412 can be optionally omitted in a step 321. Step 321 for responder 101x can then comprise a key derivation function 402c using

35 the mutually derived secret shared keys M 410, N 411, and optionally L 412 in order to mutually

derive secret shared key ke 327. An example for a key exchange step 321 is described in point 2 of section 6.2.3 of DPP specification version 1.0 on page 48.

Responder 101x can then conduct an encryption step 322, where the details for an encryption step 322 are depicted and described in connection with Figure 4c below. Step 322 can also comprise responder 101x creating or calculating a value R-auth 404, where R-auth 404 can comprise a secure hash value of the initiator nonce 317a, the second random number R-nonce 318b from a step 318 above, and public PKI keys. The value for R-auth 404 in a step 322 can comprise the value in point 3 of section 6.2.3 of DPP specification version 1.0. An encryption step 322 can comprise responder 101x encrypting the value for R-auth 404 using the mutually derived key key 327 from the third key exchange step 321 above, where the output of encryption step 322 can comprise ciphertext 322a.

Responder 101x can then conduct an encryption step 315b, where the details for an encryption step 315b are depicted and described in connection with Figure 4b below. Ciphertext 322a from an encryption step 322 in the paragraph above can also be encrypted again for an encryption step 315b. The symmetric ciphering key for encryption step 315b can comprise symmetric key k2 314c. Encryption step 315b, encryption step 322, and decryption step 320 can use a common symmetric ciphering algorithm 405a, which could comprise the Advanced Encryption Standard with Synthetic Initialization Vectors (AES-SIV) (and deciphering algorithm) also with a common set of symmetric ciphering parameters 199f. Symmetric ciphering parameters 199f can also specify the use of a block chaining mode such as cipher block chaining (CBC), counter mode (CTR), or Galois/Counter mode (GCM) and other possibilities exist as well. In addition, symmetric ciphering parameters 199f could specify a mode for message authentication, which could comprise a CMAC mode as specified in NIST publication SP-800-38B. In exemplary embodiments, a symmetric ciphering algorithm 405a can comprise the AES-SIV algorithm as specified in IETF RFC 5297. The output from an encryption step 315b can be ciphertext 315c, as depicted in Figure 4b below. As depicted in Figure 3a, the collection of steps for responder 101x comprising steps 318, 319, etc. through step 315b can comprise a step 325.

Responder 101x can then send an DPP Authentication Response message 123, where message 123 can include hash values of the bootstrap public keys H(Br 101a), H(Bi 104a), the responder ephemeral public key Pr 101e, and the ciphertext 315c. The ciphertext 315c can have encrypted values for (i) R-nonce 318b, (ii) I-nonce 317a, (iii) R-capabilities 318c, and (iv) ciphertext 322a. The ciphertext 315c can be encrypted with mutually derived symmetric encryption key k2 314c as shown for an encryption step 315b in Figure 4b. Ciphertext 322a can be encrypted with the mutually derived symmetric key ke 327a and can include a plaintext value for R-auth 404. As described above, R-auth 404 can comprise a hash value over an initiator nonce 317a received and the responder nonce 318b.

As depicted in Figure 3a, initiator 102' can receive DPP Authentication Response message 123. In exemplary embodiments, initiator 102' can conduct a step 323 to process the received responder ephemeral public key Pr 101e, such as evaluating that the received key in message 123 uses the parameters 199b specified in cryptographic parameters 199a. In addition, a step 323 can comprise initiator 102' can conduct public key validation step to ensure that received responder ephemeral public key Pr 101e is properly formatted and does not contain weak or compromising cryptographic data. For example, a step 323 could comprise initiator 102' performing the steps for an ECC Full Public-Key Validation Routine in section 5.6.2.3.2 of FIPS publication SP 800-56A (revision 2). Alternatively, step 323 can comprise initiator 102' performing the steps ECC Partial Public-Key Validation Routine in section 5.6.2.3.3 of the same FIPS publication. Other example steps within a public key validation step 323 can comprise (i) verifying the public key is not at the "point of infinity", and (ii) verifying the coordinates of the point for the public key are in the range $[0, p-1]$, where p is the prime defining the finite field. Other possibilities exist as well for evaluating and validating a received public key is cryptographically secure in a public key validation step 323, without departing from the scope of the present invention. As contemplated in the present invention, an initiator 102', a DPP server 103, or a responder 101x can conduct a public key validation step 323 each time a public key is received.

In exemplary embodiments, initiator 102' can send DPP server 103 a message 324 using secure session 302b. Message 324 can include the ID-token.device 206 (or alternatively ID.device 101b), and the ciphertext 315c. Hash values for public bootstrap keys can be useful over link 192 between responder 101x and initiator 102' in order to confirm identities, but the hash values can be substituted with an identity for device 101 in message 324 between initiator 102' and DPP server 103. The ciphertext 315c in message 324 can be the same as ciphertext 315c in message 123 in Figure 3a. In other embodiments, the hash values for the public bootstrap keys of initiator 102 and responder 101x can be included in a message 324, thereby increasing the message size by at least 64 bytes.

DPP server 103 can receive message 324 from initiator 102' and conduct a series of steps in order to process the message and reply to the initiator 102'. In exemplary embodiments, DPP server 103 can also perform a public key validation step 323 on the received responder ephemeral public key Pr 101e. Although a step 323 by DPP server 103 may appear redundant with a step 323 by initiator 102', DPP server 103 may trust its own computation for a step 323 with a higher level than relying solely on a step 323 performed by initiator 102'. Initiator 102' as operating in Figure 3a could also comprise a device with a lower level of security than DP server 103. After a step 323, DPP server 103 can record the validated, received responder ephemeral public key Pr 101e in a server database 103a, as depicted in Figure 3b below.

At step 319a, DPP server 103 can conduct a second initiator key exchange step 319a, where DPP server 103 can input the received, validated responder ephemeral public key Pr 101e and the derived initiator ephemeral private key pi 102b into an ECDH key exchange algorithm 401b in order to calculate a shared secret key N 411. The second initiator key exchange step 319a is also depicted and described in connection with Figure 4b below. Shared secret key N 411 can also be used by DPP server 103 in a step 327 below. DPP server 103 can input shared secret key N 411 into a key derivation function 402b in order to mutually derived the symmetric encryption key k2 314c.

At step 320b, DPP server 103 can use the derived key k2 314c from a step 319a in order to conduct a decryption step 320b on received ciphertext 315c. A decryption step 320b on received ciphertext 315c using key k2 314c is depicted and described in connection with Figure 4b below. DPP server 103 can read the plaintext 403b values from decryption step 320b of ciphertext 315c. The plaintext values 403b can include the initiator nonce 317a from a message 317 and message 122 above. The plaintext values 403b can also include a second ciphertext 322a (e.g. ciphertext 322a can be inside ciphertext 315c). DPP server 103 verifies that the initiator nonce 317a received in message 123 and decrypted in step 320b is equal to the initiator nonce 317a value sent in message 122 above. If the two initiator nonce 317a values are not equal, then DPP server 103 can abort the DPP procedure and subsequent steps.

At step 327, DPP server 103 can conduct a third initiator key exchange step 327, where DPP server 103 can determine the secret shared key L 412 by inputting (i) the sum or combination of the responder bootstrap public key Br 101a and the responder ephemeral public key Pr 101e, and (ii) the initiator bootstrap private key bi 104b into an ECDH key exchange algorithm 401c. A third server key exchange step 327 is depicted and described in connection with Figure 4c below. The shared secret key L 412 along with shared secret keys M 410 and N 411 can be used with a key derivation function 402c in order to output the mutually derived symmetric ciphering key ke 327a.

At step 328, DPP server 103 can conduct a decryption step 328 on the ciphertext 322a using the key ke 327a. A decryption step 328 is depicted and described in connection with Figure 4c below. DPP server 103 can use the decryption step 328 in order to read the plaintext value of R-auth 404. R-auth 404 can comprise a hash 199e value over the combination of initiator nonce 317a and the responder nonce 318b. At step 328, DPP server 103 can then also internally calculate the same value for R-auth 404, and if the internally calculated value and the received, decrypted value are equal, then the responder 101x can be consider authenticated with DPP server 103. The calculation of R-auth 404 by DPP server 103 can be according to section 6.2.3 of DPP specification version 1.0.

DPP server 103 can then conduct a step 322b, where step 322b can comprise calculating an initiator authentication value I-auth 404a. Initiator authentication value 404a can be similar to responder authentication value R-auth 404, except the values of initiator nonce 317a can be sequenced before a responder nonce 318b inside a hash 199e calculation over the two nonces. A

DPP server 103 can then encrypt the initiator authentication value 404a using an encryption step 322 using key ke 327a, as depicted in Figure 4c below. In other words, an encryption step 322 in Figure 4c depicts the encryption of R-auth 404, and encryption step 322b can comprise a DPP server 103 encrypting the initiator authentication value I-auth 404a using the key ke 327a derived in a step 327.

5 DPP server 103 can then send initiator 102' a message 329, where message 329 can include (i) an identity for device 101, such as ID-token.device 206, although ID.device 101i could be used instead, and (ii) the encrypted initiator authentication value I-auth 404a. Initiator 102' can receive message 329 over secure session 302b. In an exemplary embodiment, the value MAC.R 101d could also be within message 329, such that initiator 102' can use the MAC.R 101d to send data in
10 message 329 to responder 101x using the MAC.R 101d. Initiator 102' can determine which of a plurality of responders 101x is directed towards based on the value of ID-token.device 206 or MAC.R 101d. Initiator 102' can previously record the hash 199e values for the initiator and responder bootstrap public keys. Initiator 102 can take the recorded hash 199e values for the initiator and responder bootstrap public keys and append to them the encrypted value for I-auth 404a
15 received in message 329. Initiator 102' can send responder 101x a DPP authentication confirm message 123a, as depicted in Figure 3a. The data and steps to determine and calculate an DPP authentication confirm message 123a can be according to section 6.2.4 of DPP specification version 1.0.

Note that although the DPP authentication confirm message 123a can be equivalent to the
20 DPP authentication confirm message from section 6.2.4 of DPP specification version 1.0, an important difference can be the operation of initiator 102' in Figure 3a versus an initiator as contemplated by DPP specification version 1.0. In the embodiment of the present invention as depicted in Figure 3a, initiator 102' can record neither the responder bootstrap public key Br 101a, nor the initiator bootstrap private key bi 104b. So, using the technology depicted in Figure 3a, an
25 initiator 102' can (i) send and receive messages that are fully compatible with a DPP protocol 191, but (ii) DPP server 103 can record keys 101a and 104b. In other words, a responder 101x may not "know" that an initiator operates as either (i) an initiator 102* from Figure 1b or Figure 1c, or an initiator 102' from Figure 3a since the messages transmitted and received by the responder 101x can be fully compatible with a DPP 191.

30 Responder 101x can receive the DPP authentication confirm message 123a and process the message. Responder 101x can listen for either (i) a broadcast message with the hash 199e values for the bootstrap public keys 101a and 104a, or (ii) for a unicast message to MAC.R 101d. Responder 101x can first conduct a decryption step 328b, where decryption step 328b can use the same steps as a decryption step 328 as depicted in Figure 4c below, with a difference being decryption step 328b
35 operates on the initiator authentication value I-auth 404a instead of the R-auth 404 value.

In step 331, responder 101x can compare the decrypted value for I-auth 404a with a calculated value for I-auth 404a and if the two values match then data from the initiator 102' can be considered by responder 101x to be authenticated. After an authentication step 331, in step 332, responder 101x can take on a roll of enrollee, such that responder 101x can prepare to send a configuration attribute 330. Configuration attribute 330 can be used to specify that responder 101x can take the roll of enrollee and is expecting a set of network credentials 109 from initiator 102'. A configuration attribute 330 can be formatted and contain data fields according to section 6.3.4 of DPP specification version 1.0. For the exemplary embodiment depicted in Figure 1a, device 101 with responder 101x would prefer a role of WiFi client (or "sta") for a configuration attribute 330, but other possibilities exist as well. For the exemplary embodiment depicted in Figure 7 below a responder 101x operating within an access point 105 would prefer a role of WiFi access point (or "ap") for a configuration attribute 330.

In step 332, responder 101x can also generate a third responder random number, for an enrollee nonce E-nonce 332a. E-nonce 332a and other nonce values in Figure 3a such as the initiator nonce I-nonce 317a and responder nonce R-nonce 318b can be processed using nonce parameters 199g. Responder 101x can then conduct an encryption step 322c, where encryption step 322c can comprise (i) an encryption step 322 using key ke 327a, and (ii) the plaintext to be encrypted can be the E-nonce 332a and the configuration attribute 330. The output of an encryption step 322c can be a ciphertext 322d. As depicted in Figure 3a, the collection of steps comprising 328b through 322c for responder 101x can comprise a step 339, where a responder 101x processes both message 123a and message 123b in a step 339. Responder 101x can then send or transmit to initiator 102' a DPP Configuration Request message 123b, where the configuration request 123b can include the ciphertext 322d, where the plaintext in ciphertext 322d can include E-nonce 332a and the configuration object 330. DPP Configuration Request message 123b can be processed by responder 101x according to section 6.3.3 of DPP specification version 1.0.

Initiator 102' can receive DPP Configuration Request message 123b via connection 192 and using DPP application 102g. As contemplated herein, a DPP application 102g can perform the steps for initiator 102' depicted in Figure 3a, such as transmitting/receiving DPP 191 messages using a first MAC.I 102n for connection 192 and transmitting/receiving messages with DPP server 103 using MAC.N 102q via access network 112. Although not depicted in Figure 3a, DPP application 102g could also perform the depicted steps 323 to validate the received responder ephemeral public key Pr 101e from message 123b. For embodiments where initiator 102' uses a WAN radio 102h to connect with access network 112 and DPP server 103, then DPP application 102g could also use a different identity than MAC.N 102q to identify the physical interface of WAN radio 102h, such as an IMEI, device identity ID.device 102i, or other possibilities exist as well. For other embodiments where initiator 102' uses network access point 105 for an access network 112 to connect with DPP

server 103, then MAC.I 102n could also potentially be the same as MAC.N 102q. Other possibilities exist as well without departing from the scope of the present invention for an identity used by initiator 102' to send messages to DPP server 103.

Initiator 102' can send DPP server 103 a message 334 via secure connection 302b, where message 334 can include the ciphertext 322d. Although not depicted in Figure 3a, message 334 can also include an identity for device 101, such as, but not limited to, ID-token.device 206 or ID.device 101b, since both DPP server 103 and initiator 102' can communicate with a plurality of different devices 101 over time, and DPP server 103 may prefer to know which ciphertext 322d is associated with which device 101 in order to process the ciphertext 322d. As depicted in Figure 3a, message 334 can include the encrypted values for E-nonce 332a and configuration attribute 330 in ciphertext 322d.

DPP server 103 can receive message 334 and process the message 334. DPP server 103 can conduct a decryption step 328c using decryption step 328 depicted in Figure 4c below. A difference between decryption step 328 and decryption step 328c is that decryption step 328c uses the ciphertext 322d as input into the symmetric decryption algorithm 405b. DPP server 103 can use the previously calculated key ke 327a in order to read the plaintext within ciphertext 322d. DPP server 103 can record the plaintext E-nonce 332a and configuration attribute 330.

In step 335, DPP server 103 can process the received configuration attribute 330. For the exemplary embodiment depicted in Figure 1a, configuration attribute 330 can specify that device 101 with responder 101x may have a role of WiFi client. In exemplary embodiments, DPP server 103 can previously select a primary access network 308aa in a step 308a above, using the received networks available list 304c from initiator 102'. Also, in a step 308b above, DPP server 103 could receive or record the set of network credentials 109 associated with the selected primary access network 308aa for device 101 using device identity ID-token.device 206 (or ID.device 101i). In step 335, DPP server can process the set of network credentials 109 for primary access network 308aa into a configuration object. The configuration object with network credentials 109 can comprise a connector or configuration object as specified in section 6.3.5 and 6.3.6 of DPP specification version 1.0. The configuration object with network credentials 109 can be signed using the recorded initiator ephemeral private key pi 102b, corresponding to public key Pi 102a sent in message 317 and 122 above. Or, in another embodiment the configuration object with network credentials 109 can be signed using the recorded initiator bootstrap private key bi 104b. The steps to create and verify a signature with ECC PKI keys can be according to FIPS publication 186-4, and other possibilities exist as well.

In a step 335, DPP server 103 can process a configuration object for network credentials 109, and the configuration object can include an SSID value of SSID.network-AP 109a, a pairwise master key value of PMK.network-AP 109b, and configuration values of config.network-AP 109c. The

value of PMK.network-AP 109b can also include a value for PMKID (e.g. PMK identifier), such that PMK.network-AP 109b is used by each of a client and an access point with the PMKID value. The configuration values of config.network-AP 109c in a configuration object could also specify required supporting data for operation of device 101 with an access point 105 after a configuration step 106, such as an operating class 130a, channel list 130b, an authentication mode for device 101 to operate, which could be PSK, PMK, SAE, EAP, also with any required supporting data and/or identities. The configuration values of config.network-AP 109c in a configuration object could also include a passphrase or pre-shared key values of PSK for legacy WPA2 networks. As contemplated herein, a set of network credentials 109 can include either (i) network credentials 109 for a selected primary access network 308aa, or (ii) network credentials 109 for both a selected primary access network 308aa and a selected backup access network 308ab. Or, in another exemplary embodiment, although a single configuration object of network credentials 109 is depicted in Figure 3a, DPP server 103 could process (i) a first set of network credentials 109 for selected primary access network 308aa into a first configuration object and then (ii) a second set of network access credentials 109 for selected backup access network 308ab into a second configuration object.

At step 322e, DPP server 103 can conduct an encryption step 322e using an encryption step 322 depicted and described in connection with Figure 4c below. Encryption step 322e can be performed using key ke 327a for device 101. For encryption step 322e, the plaintext values of (i) the E-nonce 332a and (ii) the configuration object comprising network credentials 109 can be input into a symmetric encryption algorithm 405a in order to generate a ciphertext 322f. DPP server 103 can then send initiator 102' a message 338, where message 338 includes the ciphertext 322f. Message 338 can also optionally include an identity for device 101 such as ID-token.device 206 or ID.device 101i. Message 338 can be sent via secure connection 302b, as depicted in Figure 3a.

Initiator 102' using DPP application 102g can receive message 338 via secure connection 302b. Initiator 102' can use an optional identity for device 101 in a message 338 in order to determine which of a potential plurality of devices 101 associated with ciphertext 322f in order to forward the data. Initiator 102' can then send responder 101x a DPP configuration response message 124 via connection 192, where message 124 can include the ciphertext 322f, and the ciphertext 322f can include the encrypted E-nonce 332a and (i) the encrypted, signed network credentials 109.

Responder 101x can receive message 124 from initiator 102' with ciphertext 322f. Responder 101x can conduct a decryption step 328d using decryption step 328 depicted in Figure 4c below. A difference between decryption step 328 and decryption step 328d is that decryption step 328d uses the ciphertext 322f as input into the symmetric decryption algorithm 405b. Responder 101x can use the previously calculated key ke 327a (from key exchange step 321) in order to read the plaintext within ciphertext 322f. Responder 101x can confirm that plaintext in ciphertext 322f

includes the previously send E-nonce 332a. Responder 101x can read (i) the plaintext values for credentials 109. Note that credentials 109 may be previously signed by DPP server 103 using a private key initiator ephemeral private key pi 102b for the initiator 102' (or initiator bootstrap private key bi 104b), where the private key corresponds to public key Pi 102a. The responder could also verify a signature for the credentials 109 using the ephemeral public key for the initiator Pi 102a recorded with the responder 101x, such as the initiator ephemeral public key Pi 102a or the initiator bootstrap public key Bi 104a.

After reading the network credentials 109 and optionally verifying a signature using a recorded public key 102a or 104a, device 101 can complete the configuration step 106 by loading the network credentials 109 and begin operating a WiFi radio 101h with the network credentials 109, as depicted in Figure 1a. In this manner, device 101 can obtain connectivity to network access point 105. In exemplary embodiments, the initiator 102' can conduct the initiator restore step 129 depicted and described in connection with Figure 1d above, such that initiator 102' can be returned to the previous state where WiFi radio 102w operates with a user configuration 131, thus minimizing impacts of the prior initiator configuration step 127 for an initiator user 102u.

Figure 3b

Figure 3b is an illustration of an exemplary server database, with tables for a server database recording exemplary data, in accordance with exemplary embodiments. A server database 103a depicted and described above in connection with Figure 1a and Figure 3a can record data for a DPP server 103 to work or operate with an initiator 102' in order for the initiator 102' to conduct a device provisioning protocol 191 with a device 101 operating a responder 101x. A server database 103a could record a plurality of tables for a plurality of devices 101 and a plurality of initiators 102', including the depicted device credentials table 340. Other possibilities exist as well for the organization, tables, and recorded data within a device database 103a without departing from the scope of the present invention. Data within server database 103a could be encrypted using a symmetric key. As depicted in Figure 1a, a DPP server 103 operating in a network 107 could operate and record a server database 103a.

Server database 103a can record a device identity of ID-token.device 206, a responder MAC address MAC. R 101d, an initiator MAC address MAC. I 102n (or MAC.I 102q), a mode of operation for the DPP server 103 and initiator 102' comprising initiator mode 306a, a selected set of Cryptographic Parameters 199a, an Initiator Ephemeral Public Key 102a, an Initiator Ephemeral Private Key 102b, a Responder Ephemeral Public Key 101e, Bootstrap keys 399, an optional PKEX Key 198, a Shared Secret k1 314a, a Shared Secret k2 314c, and a Shared Secret ke 327a. For exemplary embodiments, not all depicted values may be recorded in a server database 103a, as

shown by the depicted value of “-“, which can mean that the value is not recorded in a server database 103a. An initiator 102’ could record the value depicted as “-“ for a server database 103a in Figure 3b for some initiator modes 306a, as depicted. For example, with the row in server database 103a depicted with a mode “Fig 5a”, the initiator 102’ could derive, record, and operate on the initiator ephemeral private key pr 102b and initiator ephemeral public key Pr 102a, and DPP server 103 may not need or record the keys. Similarly, with the row in server database 103a depicted in Figure 3b as a mode of “Fig 5b”, the initiator 102’ could derive shared secret keys k2 314c and ke 327e, while server database 103a could derive and record key k1 314a. Note that in alternative exemplary embodiments, a server database 103a could record the values M 410, N 411, and L 412 respectively instead of key k1 314a, k2 314c, and/or ke 327e, and either DPP server 103 or initiator 102’ could use the key derivation function 402a, 402b, and/or 402c, respectively in order to convert the values into keys. In exemplary embodiments, the time that a device shared secret key or a derived ephemeral PKI key, or a bootstrap key is recorded in a server database 103a can be minimized to the minimum time necessary to conduct a DPP 191. For example, shared secret keys and ephemeral PKI keys recorded in a server database 103a can be deleted once a configuration step 106 has been completed for a device 101, such as at the time DPP server 103 receives a report 605aa from device 101 in Figure 6a below.

Initiator mode 306a can specify an operating mode for initiator 102 and DPP server 103, such that a sequence of steps and transmitting/receiving PKI and/or shared secret keys can be specified by the initiator mode 306a. An initiator mode 306a could also be referred to as a “server mode” 306a, since the mode specifies the operation of a DPP server 103. Or, initiator mode 306a could be referred to as a “mode 306a”, since both steps for initiator 102’ and DPP server 103 are affected by the mode. Figure 3a above also described a first, second, third, and fourth operating initiator modes 306a. A first mode is depicted as “Fig. 3a” in Figure 3b and can comprise the mode 306a depicted in Figure 3a. Note that all depicted ephemeral PKI keys and bootstrap PKI keys, as well as all depicted shared secret keys are recorded in a server database 103a for a device 101 (except responder private keys 101b and 101f) for the first mode. In other words, when conducting a DPP 191, initiator 102’ and DPP server 103 would have all the depicted keys recorded in a server database 103a for the first mode. This first mode for an initiator mode 306a can be preferred for embodiments where (i) initiator 102’ may have a lower level of trust with network 107, or (ii) initiator 102’ could have a lower level of processing resources, or (iii) an owner for device 101 or a network 107 may prefer that responder bootstrap public key Br 101a is not shared externally from network 107, or (iv) regulatory requirements where device 101 could comprise a high value or sensitive piece of equipment. Other possibilities exist as well for the reasons different modes 306a may be used in a server database 103a to specify the steps for an initiator 102’ and DPP server 103 to conduct a DPP 191 with device 101.

A second mode is depicted as “Fig. 5a” in Figure 3b, where the steps and message flows in Figure 5a for initiator 102’ and DPP server 103 can comprise the second mode. Note that all relevant bootstrap PKI keys 399 are recorded in a server database 103a for the mode depicted in Figure 5a. The mode 306a can also have (i) the responder ephemeral public key Pr 101e and (ii) the shared secret key k1 314a recorded in a server database 103a. Note that initiator ephemeral PKI keys and secret shared keys can optionally not be recorded in a server database 103a for the second mode depicted in Figure 5a. This exemplary embodiment may be preferred for cases where (i) initiator 102’ can handle more of the cryptographic operations (e.g. reducing DPP server 103 processing load), but (ii) a network 107 and initiator user 102u could prefer that the initiator bootstrap private key bi 104b is not transmitted to or recorded in initiator 102’. An exemplary reason the second mode of “Fig 5a” could be preferred may be embodiments where multiple different devices 101 record the same initiator bootstrap public key Bi 104a, and for these embodiments a network 107 may prefer to keep the initiator bootstrap private key bi 104b recorded in a server database 103a. This embodiment was also discussed above with the devices designated with “shared keys” in Figure 2b. As depicted in Figure 3b, devices 101 with the same recorded value or number for an initiator bootstrap public key Bi 104a could also record different responder bootstrap public keys Br 101a.

A third mode 306a is depicted as “Fig. 5b” in Figure 3b, where the steps and message flows in Figure 5b for initiator 102’ and DPP server 103 can comprise the third mode. Note that all relevant bootstrap PKI keys 399 are recorded in a server database 103a for the third mode depicted in Figure 5b. The third mode 306a can be a mix between the first mode of “Fig. 3a” and the second mode of “Fig 5a”. In other words, the third mode 306a of “Fig 5b” can have the same data recorded for operation of the mode in “Fig 5a”, except the addition of recording the initiator ephemeral private key pi 102b and secret key k1 314a in server database 103a. Thus some additional computation resources are shifted from the initiator 102’ to DPP server 103 from using a mode of “Fig. 5a” to the mode of “Fig. 5b”. One reason to use the third mode of Fig. 5b could be a desire or requirement by DPP server 103 or initiator user 102u or a device 101 owner to have the cryptographic operations of a DPP 191 more evenly shared between an initiator 102’ and a DPP server 103. In other words, a mode depicted in Fig. 3a could comprise a mode that is mostly on DPP server 103, a mode depicted in Figure 5a could comprise a mode that is mostly on initiator 102’, and a mode depicted in Figure 5b could comprise a balanced mode where a mix of the keys and cryptographic operations are more evenly shared between DPP server 103 and initiator 102’.

A fourth mode 306a in Figure 3b can comprise initiator 102’ operating as an initiator 102* depicted in Figure 1b and Figure 1c, and in this case initiator 102* can record the keys shown for the fourth mode 306a in Figure 3b. A DPP server 103 can still provide the bootstrap keys 399 shown to initiator 102* in this fourth mode of operation, or the initiator 102* could obtain the bootstrap keys 399 from other means, such as “out of band” messages 121 and 126 in Figure 1c. In addition, an

initiator 102* could obtain the bootstrap keys “in band” (e.g. via WiFi link 192) using PKEX key 198.

Bootstrap keys 399 in a server database 103a can record the bootstrap keys depicted in Figure 3b, which could comprise a responder bootstrap public key Br 101a, an initiator bootstrap public key Bi 104a, and an initiator bootstrap private key 104b. For embodiments where a device 101 can support multiple different sets of cryptographic parameters 199a, a server database 103a can record the selected set of cryptographic parameters 199a for communication between an initiator 102' and device 101 with responder 101x. The keys for “bootstrap keys 399” can be according to the selected set of cryptographic parameters 199a. In other words, a device database 104 in Figure 2b above can record all versions of the bootstrap PKI keys for all supported cryptographic parameters 199a for a device 101. However, since a set of cryptographic parameters 199a will need to be selected for a DPP 191 (e.g. one of the responder bootstrap public keys Br 101a selected), then the bootstrap keys 399 for that selected set of cryptographic parameters 199a can be recorded in a server database 103a.

A server database 103a can also record a preshared PKEX key 198, if PKEX is optionally supported by responder 101x. PKEX key 198 can be used by initiator 102' to securely transfer bootstrap public keys in band (i.e. via connection 192 in Figure 1a and Figure 3a). The use of a PKEX protocol to exchange the public keys is described in section 5.6 of DPP specification version 1.0. For embodiments where initiator 102' uses PKEX key 198 with responder 101x, the initiator bootstrap private key bi 104b preferably continues to reside within a network 107 and can be in a server database 103a. PKEX key 198 can be used with responder 101x MAC address MAC.R 101d. Embodiments can omit the use of PKEX key 198, as depicted with the value of “NA” for the key in Figure 3b. Derived, shared secret keys with responder 101x can also be recorded in a server database 103a. The derived shared secret keys could comprise key k1 314a, key k2 314c, and key ke 327e. The steps to derive and use these keys are depicted and described below in Figure 4a, Figure 4b, and Figure 4c, respectively. As discussed above, some keys may reside in server database 103a, and other keys may reside within an initiator 102' (depicted with a “-”), depending on the mode 306a.

The present invention contemplates other implementations of a device provisioning protocol besides the DPP specification version 1.0. These other implementations could have different mutually derived shared secret keys than those depicted in Figure 3b, and the different mutually derived shared secret keys could be recorded in a server database as well. For example, a different key derivation function 402 could be used to derive a secret shared key than that specified in DPP specification version 1.0. In addition, other or different PKI keys for a responder 101x and/or initiator 102' could be recorded in a server database 103a. For example, the DPP specification version 1.0 could be extended to explicitly include a DPP server 103 public and private key, in order

to extend support with a DPP server 103, and in these embodiments the DPP server 103 public and private keys could be included in a server database 103a. Other possibilities exist as well for (i) a server database 103a to record PKI keys and shared secret keys to support (ii) a device provisioning protocol without departing from the scope of the present invention.

5 A server database 103a could also record a device credentials table 340. Device credentials table 340 can record a device identity ID-token.device 206 (or ID.device 101i), a selected primary network 308aa, a selected backup network 308ab, a set of primary network credentials 109, and a set of backup network credentials 109. The selected primary network 308aa and selected backup network 308ab could be determined in a step 308a in Figure 3a by DPP server 103 or network 107.
10 The set of primary network credentials 109 can comprise the set of credentials (plus configuration) for a device 101 to use with the selected primary network 308aa. The set of backup network credentials 109 can comprise the set of credentials (plus configuration) for a device 101 to use with the selected backup network 308ab. The credentials 109 in a device credentials table 340 could be obtained from during a step 308b as depicted and described above in connection with Figure 3b.

15 For example, with the first device depicted in a device credentials table 340, device 101 could record a selected primary network 308aa for network access point 105 in Figure 1a. The value recorded for 308aa could be SSID.network-AP 109a. The credentials in device table for the first device could comprise the credentials 109 depicted in Figure 1a. The depicted designation of “109a-1” means the first instance of a value of SSID.network-AP 109a. The device in the second row can
20 use the same value for SSID.network-AP 109a. The device in the last depicted row can use a different value for a selected primary network 308aa of “109a-3” which could be the third distinct value in device credentials table 340. Values of “304c-1” for selected primary network 308aa can mean a first value from a networks identity list 304c from Figure 3a, values of “304c-2” can mean a second value from a networks available list 304c, etc.

25 For embodiments where a network other than network 107 is selected for device 101 to be configured with credentials, the credentials in a device credentials table 340 are depicted as “from ...”, where “...” can be the depicted, selected network from a networks available list 304c. In exemplary embodiments, the credentials for networks external to network 107 can be encrypted by the external network using Elgamal encryption and either (i) the responder bootstrap public key Br
30 101a or (ii) the responder ephemeral public key Pr 101e. The DPP server 103 could send either (i) the responder bootstrap public key Br 101a or (ii) the responder ephemeral public key Pr 101e to the external network for the external network to encrypt the credentials using the public key Br 101a or Pr 101e and an Elgamal asymmetric encryption algorithm, which was also described above. Thus, the credentials are stored as ciphertext in a server database 103a and only the device 101 with
35 responder 101x could feasibly decrypt the ciphertext in order to read the credentials for networks external to network 107. Additional details for receiving credentials from an external network, and

recording the encrypted credentials in a server database 103a are depicted and described below in connection with Figure 5a, depicted as credentials 109'.

Note the present invention contemplates that (a) a selected access network 112 for device 101 in a step 308a above in Figure 3a by a DPP server 103 or network 107 could (b) utilize wireless networking technology other than WiFi or 802.11 based technology. For example, surrounding wireless networks for device 101 could use exemplary potential use of 5G wireless technology or LPWAN networking technology depicted in a networks available list 304c in Figure 3a above for potential access networks 112 for device 101. Or, a WiFi access point could also be selected by a DPP server 103 or a network 107 for device 101 in step 308a above in Figure 3a using the networks available list 304c.

For the exemplary embodiment depicted in Figure 3a and also Figure 1a above, the exemplary network access point 105 can be selected in a step 308a using SSID.network-AP 109a from Figure 1a, although different networks in a networks available list 304c could be selected instead, based on exemplary criteria listed for a step 308a above in Figure 3a. In addition, a network access point 105 in Figure 1a can comprise a form of an access network 112 as contemplated herein, such that device 101 obtains access to IP network 113 or the globally routable Internet via an exemplary network access point 105. A selected access network 112 for device 101 in a step 308a does not need to belong to network 107 from Figure 1a, and could belong to a different network (so long as network 107 could obtain or point to credentials 109 for the access network 112 in order to conduct the configuration step 106 from Figure 1a). Further, a step 308a could comprise the selection of both a primary access network 112 for device 101 and a second, backup access network 112 for device 101. The use of a primary and backup network for device 101 can provide increased reliability for the operation of device 101, such that if the primary access network 112 is not available then device 101 could use the selected backup network in order for device 101 to maintain connectivity to an IP network 113.

Figure 4a

Figure 4a is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key, and for using the derived shared secret key to encrypt and decrypt data, in accordance with exemplary embodiments. Exemplary steps for an initiator 102' and a responder 101x to mutually derive keys can comprise (i) a key exchange 314 step for DPP server 103 or initiator 102' (depending on the mode 306a) to derive a symmetric encryption key k1 314a and (ii) a key exchange 319 step for a responder 101x to derive the same symmetric encryption key k1 314a. Exemplary steps for a responder 101x to decrypt data and initiator 102' for DPP server 103 (depending on a mode 306a) to encrypt data can comprise (a) an

5 encryption step 315 for initiator 102' or DPP server 103 (depending on a mode 306a) to utilize the symmetric encryption key k1 314a to convert plaintext to ciphertext, and (b) a decryption 320 step for responder 101x to utilize the symmetric encryption key k1 314a to convert the ciphertext received from initiator 102' into plaintext. The use of the steps for a key exchange 314, a key
10 exchange 319, encryption 315, and decryption 320 were also depicted and described in connection with Figure 3a above. Additional detail regarding the use of these steps will be provided herein with Figure 4a. Further, although the steps are depicted specifically for the use of particular keys and plaintext/ciphertext combinations in Figure 4a, the steps illustrated can be used with different keys and plaintext/ciphertext combinations. For example, Figure 4b below depicts the same functionality
15 used by the nodes as in Figure 4a, but with (i) different PKI keys input to mutually derive a different symmetric encryption key, (ii) a different key derivation function and (iii) different plaintext encrypted and different ciphertext decrypted.

The processes and operations, described below with respect to all of the logic flow diagrams and flow charts may include the manipulation of signals by a processor and the maintenance of these
20 signals within data structures resident in one or more memory storage devices. For the purposes of this discussion, a process can be generally conceived to be a sequence of computer-executed steps leading to a desired result.

These steps usually require physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic, or optical signals capable of
25 being stored, transferred, combined, compared, or otherwise manipulated. It is convention for those skilled in the art to refer to representations of these signals as bits, bytes, words, information, elements, symbols, characters, numbers, points, data, entries, objects, images, files, or the like. It should be kept in mind, however, that these and similar terms are associated with appropriate physical quantities for computer operations, and that these terms are merely conventional labels
30 applied to physical quantities that exist within and during operation of the computer.

It should also be understood that manipulations within the computer are often referred to in terms such as listing, creating, adding, calculating, comparing, moving, receiving, determining, configuring, identifying, populating, loading, performing, executing, storing etc. that are often
35 associated with manual operations performed by a human operator. The operations described herein can be machine operations performed in conjunction with various input provided by a human operator or user that interacts with the device, wherein one function of the device can be a computer.

In addition, it should be understood that the programs, processes, methods, etc. described herein are not related or limited to any particular computer or apparatus. Rather, various types of general purpose machines may be used with the following process in accordance with the teachings
described herein.

The present invention may comprise a computer program or hardware or a combination thereof which embodies the functions described herein and illustrated in the appended flow charts. However, it should be apparent that there could be many different ways of implementing the invention in computer programming or hardware design, and the invention should not be construed
5 as limited to any one set of computer program instructions.

Further, a skilled programmer would be able to write such a computer program or identify the appropriate hardware circuits to implement the disclosed invention without difficulty based on the flow charts and associated description in the application text, for example. Therefore, disclosure of a particular set of program code instructions or detailed hardware devices is not considered
10 necessary for an adequate understanding of how to make and use the invention. The inventive functionality of the claimed computer implemented processes will be explained in more detail in the following description in conjunction with the remaining Figures illustrating other process flows.

Further, certain steps in the processes or process flow described in all of the logic flow diagrams below must naturally precede others for the present invention to function as described.
15 However, the present invention is not limited to the order of the steps described if such order or sequence does not alter the functionality of the present invention. That is, it is recognized that some steps may be performed before, after, or in parallel other steps without departing from the scope and spirit of the present invention.

The processes, operations, and steps performed by the hardware and software described in
20 this document usually include the manipulation of signals by a CPU or remote server and the maintenance of these signals within data structures resident in one or more of the local or remote memory storage devices. Such data structures impose a physical organization upon the collection of data stored within a memory storage device and represent specific electrical or magnetic elements. These symbolic representations are the means used by those skilled in the art of computer
25 programming and computer construction to most effectively convey teachings and discoveries to others skilled in the art.

A key exchange 314 step for initiator 102' or DPP server 103 (depending on a mode 306a) to derive a symmetric encryption key 314a can utilize a selected set of cryptographic parameters 199a as depicted and described in connection with Figure 2b above. As depicted in Figure 4a, a key
30 exchange algorithm 401a can receive input both of a responder bootstrap public key Br 101a and an initiator ephemeral private key pi 102b. The key exchange algorithm 401a could comprise a Diffie Hellman key exchange (DH), an Elliptic Curve Diffie Hellman key exchange (ECDH), and other possibilities exist as well without departing from the scope of the present invention. A key exchange algorithm 401a can support either PKI keys based on elliptic curves or RSA algorithms, although
35 support of elliptic curves may be preferred in some exemplary embodiments due to their shorter key lengths and lower processing requirements.

A summary of ECDH as a key exchange algorithm 401a is included in the Wikipedia article titled "Elliptic Curve Diffie-Hellman" from March 9, 2018, which is herein incorporated by reference. An exemplary embodiment of key exchange algorithm 401a could comprise a "One-Pass Diffie-Hellman, C(1, 1, ECC CDH)" algorithm as described in section 6.2.2.2 on page 99 of the National Institute of Standards and Technology (NIST) document "NIST SP 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" from April, 2018 which is hereby incorporated by reference its entirety. Other key exchange algorithms in NIST SP 800-56A could be utilized as well for a key exchange algorithm 401a in Figure 4a, Figure 4b, and Figure 4c without departing from the scope of the present invention. Example calculations for an ECDH key exchange for a key exchange algorithm 401a are shown on pages 88 and 89 of DPP specification version 1.0, where a selected set of parameters 199a can use the ECC curve P-256 with PKI key lengths of 256 bits.

Other algorithms to derive a shared symmetric encryption key using public keys and private keys may also be utilized in a key exchange algorithm 401a (and 401b in Figure 4b and 401c in Figure 4c), such as, but not limited to, the American National Standards Institute (ANSI) standard X-9.63. Cryptographic parameters 199a can also include information for using a key exchange algorithm 401a and a key derivation function 402a in order to derive a commonly shared symmetric encryption key k1 314a. As contemplated herein, the terms "selected set of cryptographic parameters 199a" and "cryptographic parameters 199a", and "parameters 199a" can be equivalent. Parameters 199a input into a key exchange algorithm 401a can include a time-to-live for a key k1 314a that is derived, a supported point formats extension, where the supported point formats extension could comprise uncompressed, compressed prime, or "compressed char2" formats, as specified in ANSI X-9.62. In other words, an ECC keys input into a key exchange algorithm 401a may have several different formats and a set of parameters 199a can be useful to specify the format.

As depicted in Figure 4a, the output of a key exchange algorithm 401a, such as an ECDH key exchange, can comprise a secret shared key M 410. The secret shared key M 410 can be input into both (i) a first key derivation function 402a in order to calculate the key k1 314a and (ii) a second key derivation function 402c in order to calculate a symmetric encryption key ke 327a as depicted in Figure 4c below. Key derivation function 402a could use a hash algorithm 199e from parameters 199a, where an input into the hash algorithm 199e can comprise the value M 410. As contemplated herein, a secret shared key can also be referred to as a value, since the secret shared key can be a numeric value or a number in the form of a sequence of digits. In DPP specification version 1.0, a key derivation function 402a to determine key k1 314a is depicted on the fourth line of page 48. Other possibilities exist as well for a key derivation function 402a without departing from the scope of the present invention. For example, with other embodiments besides those within DPP specification version 1.0, the output of a key derivation function 402a can be both a symmetric

encryption key k1 314a and a message authentication code (MAC) key, where a MAC key can function to verify message integrity of ciphertexts generated with the symmetric encryption key 314a.

As depicted in Figure 4a, an encryption 315 step could be utilized with the derived symmetric encryption key k1 314a in order to convert plaintext 403b into ciphertext 315a. An encryption 315 step can utilize a symmetric encryption algorithm 405a, where symmetric encryption algorithm 405a could comprise an algorithm according to the Advanced Encryption Standard (AES), Blowfish, and Triple Data Encryption Standard (3DES), and other possibilities exist as well. As depicted in Figure 4a, a symmetric encryption algorithm 405a can comprise the AES-SIV algorithm as specified in IETF RFC 5297. Parameters 199f could specify settings for a symmetric ciphering algorithm 405a as well, such as key length and block cipher modes such as, but not limited to, ECB, CBC, OFB, and CFB. Parameters 199f may also specify an authenticated encryption mode such as CCM (NIST SP800-38C), GCM (NIST SP800-38D), and other examples exist as well.

Symmetric encryption algorithm 405a can accept input of (i) plaintext 403b, (ii) symmetric encryption key k1 314a, (iii) parameters 199f, and (iv) optionally an initialization vector (not shown in Fig. 4a), and output ciphertext 315a. Although not depicted in Figure 4a, an encryption 405a step can output a message authentication code (MAC), where the MAC can be calculated with a MAC key from a key derivation function 402a. In exemplary embodiments the use of a MAC and MAC key, as well as an initialization vector can be optionally omitted. The exemplary listed values for plaintext and ciphertext in step 315 were described in connection with depicted step 315 in Figure 3a.

Key exchange 319 step for responder 101x depicted in Figure 4a can correspond to key exchange 314 step by an initiator 102' or DPP server 103 (depending on a mode 306a). Key exchange 319 step can comprise inputting or using the initiator ephemeral Public key Pi 102a (from message 122) and the responder bootstrap private key br 101b into key exchange algorithm 401a, which can comprise the same or equivalent key exchange algorithm 401a depicted and described in connection with key exchange step 314 described above. Other elements or algorithms within a key exchange step 319 can be equivalent to a key exchange step 314 above, including the use of shared parameters 199a. In this manner, initiator 102' or DPP server 103 (depending on a mode 306a) and responder 101x can securely derive the same symmetric encryption key k1 314a as depicted in Figure 4a.

A decryption 320 step allows responder 101x to convert the ciphertext 315a received in a message 122 from Figure 3a into plaintext 403a. Decryption 320 step can utilize a symmetric decryption algorithm 405b, which could comprise the same algorithm used in symmetric encryption algorithm 405a except the algorithm being used for decryption instead of encryption. For example, if symmetric encryption algorithm 405a in Figure 4a comprises an AES-SIV encryption, then

symmetric decryption algorithm 405b can comprise an AES-SIV decryption. Note that the same values are input into symmetric decryption algorithm 405b as symmetric encryption algorithm 405a, such as symmetric encryption key k1 314a and parameters 199f in order to convert ciphertext 315a back into plaintext 403a. Responder 101x can then read and process plaintext 403a after a decryption
5 320 step.

For embodiments where a MAC code is received with ciphertext 315a, responder 101x can also verify the MAC code using a MAC key from a key derivation function 402a in a step 319 in order to verify the integrity of ciphertext 315a. Note that AES-SIV and similar symmetric ciphering algorithms can have a message integrity check built into the algorithm, and thus a separate MAC
10 code and MAC key can be optionally omitted in some exemplary embodiments. In exemplary embodiments, the successful decryption of a ciphertext into a plaintext using decryption algorithms 405b supports mutual authentication between the two nodes, since successful decryption can only take place when the two nodes record the depicted private keys used for the depicted key exchanges 314 and 319.

Figure 4b

Figure 4b is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key, and for using the derived shared secret key to
20 encrypt and decrypt data, in accordance with exemplary embodiments. Exemplary steps for (i) an initiator 102' or a DPP server 103 (depending on the mode 306a) and (ii) a responder 102x to mutually derive keys can comprise (a) a key exchange 319a step for DPP server 103 or initiator 102' (depending on the mode 306a) to derive a symmetric encryption key k2 314c and (b) a key exchange 314b step for a responder 101x to derive the same symmetric encryption key k2 314c. Exemplary
25 steps for a responder 101x to encrypt data and initiator 102' to decrypt data can comprise (i) an encryption 315b step for responder 101x to utilize the symmetric encryption key k2 314c to convert plaintext to ciphertext, and (ii) a decryption 320b step for either initiator 102' or DPP server 103 (depending on the mode 306a) to utilize the symmetric encryption key k2 314c to convert the ciphertext received from responder 101x into plaintext. The use of the steps for a key exchange
30 314a, a key exchange 319a, encryption 315b, and decryption 320b were also depicted and described in connection with Figure 3a above.

Note that steps in Figure 4b and the steps in Figure 4a can share some algorithms and values, and the descriptions for the algorithms and values in Figure 4a can be applicable for Figure 4b. For example, a same key exchange algorithm 401a depicted and described in Figure 4a can also be used
35 in Figure 4b. The key exchange algorithm 401a can comprise an ECDH key exchange. The set of parameters 199a depicted and described in Figure 4a can also be used in Figure 4b. Further, a

symmetric encryption algorithm 405a and a symmetric decryption algorithm 405b can be the same in Figure 4a and Figure 4b. A set of parameters 199f for a symmetric encryption algorithm 405a and a symmetric decryption algorithm 405b can be the same in Figure 4a and Figure 4b

A responder 101x can conduct a key exchange step 314b. In a key exchange step 314b, the initiator ephemeral public key P_i 102a received in message 122 from Figure 3a can be input into an ECDH key exchange algorithm 401a along with the responder ephemeral private key pr 101f derived in a step 318 in Figure 3a in order to output a secret shared key N 411. The value for shared secret key N 411 can be recorded and also used in the derivation of a key ke 327a in a key exchange step 321 below in Figure 4c. For a step 314b, the value for the shared secret key N 411 can be input into a key derivation function 402b in order to derive a mutually shared symmetric key k_2 314c. In exemplary embodiments which support DPP specification version 1.0, a key derivation function 402b can comprise the key derivation function for key k_2 in point 2 of section 6.2.3 of DPP specification version 1.0.

Either an initiator 102' or a DPP server 103 can conduct a key exchange step 319a, depending on a mode 306a from a server database 103a. At step 319a, a received, validated responder ephemeral public key Pr 101e and a derived initiator ephemeral private key pi 102b can be input into an ECDH key exchange algorithm 401a in order to calculate a shared secret key N 411. Shared secret key N 411 can also be used by initiator 102' or DPP server 103 (depending on a mode 306a) in a step 327 below. DPP server 103 can input shared secret key N 411 into a key derivation function 402b in order to mutually derived the symmetric encryption key k_2 314c. In exemplary embodiments which support DPP specification version 1.0, a key derivation function 402b can comprise the key derivation function for key k_2 in section 6.2.4 of DPP specification version 1.0.

Responder 101x can conduct an encryption step 315b, where the use for an encryption step 315b are depicted and described in connection with Figure 3a above. Ciphertext 322a can comprise a ciphertext from an encryption step 322 below in Figure 4c. In other words, a responder 101x can conduct the encryption step 322 below and also key exchange step 321 below in Figure 4c before conducting the encryption step 315b. As depicted in Figure 4b, the ciphertext 322a can also be encrypted again for an encryption step 315b. The symmetric ciphering key for encryption step 315b can comprise symmetric key k_2 314c from a key exchange step 314b in the paragraph above. The output from an encryption step 315b can be ciphertext 315c, as depicted in Figure 4b.

A decryption 320b step can be performed by either an initiator 102' or a DPP server 103, depending on a mode 306a recorded in a server database 103a as depicted in Figure 3b. A decryption 320b step converts the ciphertext 315c received in a message 123 from Figure 3a into plaintext 403b. Decryption 320b step can utilize a symmetric decryption algorithm 405b, which could comprise the same algorithm used in symmetric encryption algorithm 405a except the algorithm being used for decryption instead of encryption. Note that the same values are input into

symmetric decryption algorithm 405b as symmetric encryption algorithm 405a, such as symmetric encryption key k2 314c and parameters 199f in order to convert ciphertext 315c back into plaintext 403b. Initiator 102' or a DPP server 103 (depending on a mode 306a) can read and process plaintext 403b after a decryption 320b step. In exemplary embodiments, the successful decryption of a ciphertext into a plaintext using decryption algorithms 405b supports mutual authentication between the two nodes, since successful decryption can only take place when the two nodes record the depicted private keys used for the depicted key exchange.

Figure 4c

Figure 4c is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key, and for using the derived shared secret key to encrypt and decrypt data, in accordance with exemplary embodiments. Exemplary steps for (i) an initiator 102' or a DPP server 103 (depending on the mode 306a) and (ii) a responder 102x to mutually derive keys can comprise (a) a key exchange 327 step for DPP server 103 or initiator 102' (depending on the mode 306a) to derive a symmetric encryption key ke 327a and (b) a key exchange 321 step for a responder 101x to derive the same symmetric encryption key ke 327a. Note that in some exemplary embodiments, such a mode 306a depicted and described in connection with Figure 5a and Figure 5b below, (i) a key exchange algorithm 401c can be processed by a DPP server 103 a key derivation function 402c can be processed by an initiator 102'. In other words, for some embodiments the key exchange 327 step can be shared or distributed between a DPP server 103 and an initiator 102'.

Exemplary steps in Figure 4c for a responder 101x to encrypt data and initiator 102' to decrypt data can comprise (i) an encryption 322 step for responder 101x to utilize the symmetric encryption key ke 327a to convert plaintext to ciphertext, and (ii) a decryption 328 step for either initiator 102' or DPP server 103 (depending on the mode 306a) to utilize the symmetric encryption key ke 327a to convert the ciphertext received from responder 101x into plaintext. The use of the steps for a key exchange 321, a key exchange 327, encryption 322, and decryption 328 were also depicted and described in connection with Figure 3a above.

Note that steps in Figure 4c and the steps in Figure 4a can share some algorithms and values, and the descriptions for the algorithms and values in Figure 4a can be applicable for Figure 4c. For example, the key exchange algorithm 401c can comprise an ECDH key exchange. The set of parameters 199a depicted and described in Figure 4a can also be used in Figure 4c. Further, a symmetric encryption algorithm 405a and a symmetric decryption algorithm 405b can be the same in

Figure 4a and Figure 4c. A set of parameters 199f for a symmetric encryption algorithm 405a and a symmetric decryption algorithm 405b can be the same in Figure 4a and Figure 4c.

A responder 101x can conduct a key exchange step 321. In a key exchange step 321, (i) the initiator bootstrap public key B_i 104a received in message 126 from Figure 1c can be input into an ECDH key exchange algorithm 401c along with (ii) both the responder ephemeral private key pr 101f and the responder bootstrap private key br 101b. An ECDH key exchange algorithm 401c can use $(pr\ 101f + br\ 101b)$ modulo q , as specified in DPP specification version 1.0 in point 2 of section 6.2.3, so algorithm 401c for responder 101x can be slightly different than algorithm 401a for responder 101x above in Figure 4a and Figure 4b. The output of key exchange algorithm 401c can be a shared secret key L 412.

The responder 101x can then conduct a key derivation function 402c in order to calculate a shared secret key ke 327a. Note that for a key derivation function 402c, the input includes all of key M 410 from a step 314, key N from a step 314b, and the derived key L 412 from step 321. As contemplated herein, a key derivation function 402a, 402b, and 402c can also use other data mutually shared between (i) responder 101x and (ii) initiator 102' or DPP server 103 (depending on a mode 306a). In exemplary embodiments which support DPP specification version 1.0, a key derivation function 402c can comprise the key derivation function for key ke 327a in point 2 of section 6.2.3 of DPP specification version 1.0.

Either an initiator 102' or a DPP server 103 can conduct a key exchange step 327, depending on a mode 306a from a server database 103a and transmitted to DPP server 103 in a message 311 as depicted in Figure 3a. At step 327, (i) a combination of a recorded responder bootstrap public key Br 101a and received responder ephemeral public key Pr 101e, and (ii) the recorded initiator bootstrap private key bi 104b can be input into an ECDH key exchange algorithm 401c in order to calculate a shared secret key L 412. Shared secret key L 412 can be input into a key derivation function 402c, along with shared secret key M 410 from a step 319 and shared secret key N 411 from a step 319a in order to mutually derive the symmetric encryption key ke 327a. In exemplary embodiments which support DPP specification version 1.0, a key derivation function 402c can comprise the key derivation function for key ke 327e in section 6.2.4 of DPP specification version 1.0.

Responder 101x can conduct an encryption step 322, where the use for an encryption step 322 are depicted and described in connection with Figure 3a above and also Figure 4b above. Plaintext 403c in a step 322 can comprise the responder authentication value 404, where value R_{auth} 404 can comprise a secure hash value with input including the initiator nonce 317a, the second random number R_{nonce} from a step 318 above in Figure 3a, and public PKI keys. The value for R_{auth} 404 in a step 322 (also described with step 322 in Figure 3a) can comprise the value in point 3 of section 6.2.3 of DPP specification version 1.0. The symmetric ciphering key for encryption step

322 can comprise symmetric key ke 327a from a key exchange step 321. The output from an encryption step 322 can be ciphertext 322a, as depicted in Figure 4c. Note that ciphertext 322a in a step 322 is also encrypted again in a step 315b above in Figure 4b. In other words, a responder 101x can conduct the encryption step 322 herein for Figure 4c (along with the key exchange 321 herein in Figure 4b) before conducting the encryption step 315b in Figure 4b above.

A decryption step 328 can be performed by either an initiator 102' or a DPP server 103, depending on a mode 306a recorded in a server database 103a as depicted in Figure 3b. A decryption 328 step converts the ciphertext 322a received in a message 123 (after decrypting a first layer of encryption of ciphertext 315b in a step 320b) from Figure 3a into plaintext 403c. Decryption 328 step can utilize a symmetric decryption algorithm 405b, which could comprise the same algorithm used in symmetric encryption algorithm 405a except the algorithm being used for decryption instead of encryption. Note that the same values are input into symmetric decryption algorithm 405b as symmetric encryption algorithm 405a in Figure 4c, such as symmetric encryption key ke 327a and parameters 199f in order to convert ciphertext 322a back into plaintext 403c. Initiator 102' or a DPP server 103 (depending on a mode 306a) can then read and process plaintext 403c after a decryption 328 step. In exemplary embodiments, the successful decryption of a ciphertext into a plaintext using decryption algorithms 405b supports mutual authentication between the two nodes, since successful decryption of ciphertext 322a can only take place when the two nodes record the depicted private keys used for the depicted key exchange.

As depicted and described in connection with Figure 3a, a responder 101x can also conduct both an encryption step 322 and a decryption step 328. The steps for responder 101x to conduct a decryption step 328 can comprise step 328b and step 328d as depicted and described in Figure 3a. When responder 101x conducts decryption step 328 using symmetric encryption key ke 327a, the ciphertext and plaintext will comprise different values than those depicted in Figure 4c. For example, in Figure 3a a responder 101x can use a decryption step 328b (e.g. step 328 but with different ciphertext than that depicted in Figure 4c) in order to decrypt the encrypted initiator authentication value 404a into a plaintext authentication value 404a. Also, in Figure 3a a responder 101x can use a decryption step 328d (e.g. step 328 but with different ciphertext 322f than that depicted in Figure 4c) in order to decrypt the encrypted network credentials 109.

Note that in some exemplary embodiments, the "plaintext" network credentials 109' (for the purposes of a step 328d in Figure 5a) can comprise network credentials 109' that can also be encrypted with an asymmetric ciphering algorithm such as Elgamal using a step 336 in Figure 5a below. In other words, network credentials 109' can comprise credentials that are "double encrypted" with both (i) a symmetric key 327a (which can be decrypted by responder 101x using a decryption step 328d, where decryption step 328d uses the depicted decryption 328 in Figure 4c), and (ii) an asymmetric ciphering algorithm such as Elgamal from a step 336, (which could be

decrypted by responder 101x using a step 337 from Figure 5a below and either (i) responder ephemeral private key pr 101f or (ii) responder bootstrap private key br 101b).

As depicted and described in connection with Figure 3a, a DPP server 103 or initiator 102' (in Figure 5a) can conduct both an encryption step 322 and a decryption step 328. The steps for a DPP server 103 or initiator 102' (depending on a mode 306a) to conduct an encryption step 322 can comprise step 322b and step 322e as depicted and described in Figure 3a. When DPP server 103 or initiator 102' conducts encryption step 322b or 322e using symmetric encryption key ke 327a, the ciphertext and plaintext will comprise different values than those depicted in Figure 4c. For example, in Figure 3a a DPP server 103 can use an encryption step 322b (e.g. step 322 but with different plaintext than that depicted in Figure 4c) in order to encrypt the initiator authentication value 404a. In Figure 5a, an initiator 102' can use an encryption step 322b (e.g. step 322 but with different plaintext than that depicted in Figure 4c) in order to encrypt the initiator authentication value 404a. In Figure 3a a DPP server 103 can use an encryption step 322e (e.g. step 322 but with different plaintext than that depicted in Figure 4c) in order to encrypt the network credentials 109. In Figure 5a, an initiator 102' can use an encryption step 322e (e.g. step 322 but with different plaintext than that depicted in Figure 4c) in order to encrypt the network credentials 109'.

Figure 5a

Figure 5a is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device database, a DPP server, an initiator proxy, and a responder, in accordance with exemplary embodiments. Before starting steps and message flows depicted in Figure 5a, the initiator, DPP server, and device database depicted for system 500 in Figure 5a may previously complete exemplary message flows and steps depicted for a step 301 in Figure 3a above. System 500 can include a device database 104, DPP server 103, initiator proxy 102', and responder 101x operating in device 101. Initiator proxy 102' can communicate with DPP server 103 via IP network 113 using an access network 112, also as depicted in Figure 3a above. As contemplated herein, an "initiator proxy 102'" depicted in Figure 5a may also be referred to as initiator 102', and an initiator 102' and initiator proxy 102' may be equivalent. A difference between system 500 in Figure 5a and system 300 in Figure 3a can be that (i) responder bootstrap public key Br 101a, (ii) initiator bootstrap public key Bi 104a, (iii) initiator ephemeral public key Pi 102a, and (iv) initiator ephemeral private key pi 102b are recorded and operated within an initiator 102'. In exemplary embodiments depicted in Figure 5a the initiator bootstrap private key bi 104b can remain recorded in a DPP server 103 and not exposed to initiator 102'. The exemplary embodiment depicted in Figure 5a can comprise a second mode 306a of operation for initiator 102' and DPP server 103, as depicted in Figure 3b above.

At step 502, DPP server 103 can select a mode 306a for both initiator 102' and DPP server 103. Note that the mode 306a selected is not normally observable to a responder 101x, since the full and complete set of DPP 191 messages between responder 101x and initiator 102' can be transmitted and received. In other words, (a) the different distribution of PKI and shared secret keys in a server database 103a and initiator 102' for a first mode 306a in Figure 3a and a second mode 306a in Figure 5a (b) would not normally be observable to a responder 101x. The recording of PKI and shared secret keys in a server database is depicted in a server database 103a in Figure 3b, where the value "--" can mean that the key is recorded and/or operated by initiator 102'. Step 502 can also comprise DPP server 103 selecting an initiator bootstrap public key Bi 104a and a responder bootstrap public key Br 101a for device 101 using ID-token device 206.

In exemplary embodiments, a responder 101x for a device 101 could record a plurality of responder bootstrap public keys Br 101a, as depicted in Figure 2a above. Reasons could include the use of different keys Br 101a for different purposes, such as a first key Br 101a for a responder authentication 141 and a second key Br 101a for a mutual authentication 142. Or, a third key Br 101a could be recorded or mapped in a tag 101c, while a fourth key Br 101a in a responder 101x could only be recorded in a network 107. Responder 101x can designate the use of the fourth key Br 101a with a higher level of trust, with the following exemplary results (A) and (B). For (A), device 101 could provide additional privileges for a configuration step 106 for an initiator 102' or initiator 102* that uses the second key Br 101a (e.g. the first key Br 101a could be easier for potentially any initiator 102' to obtain if the first key Br 101a is recorded in a QR code for device 101). As one example of additional privileges, a set of credentials 109 delivered to device 101 using the fourth key Br 101a could allow the installation of a new certificate authority root certificate for device 101. As one example of normal or other privileges, a set of credentials 109 delivered to device 101 using the third key Br 101a could deny the installation of a new certificate authority root certificate for device 101 as well.

In summary for a step 502, (i) a device 101 with responder 101x can use a plurality of responder bootstrap public keys Br 101a (including for reasons other than simply use of different sets of cryptographic parameters 199a for the different keys), and (ii) a DPP server 103 can select the responder bootstrap public key Br 101a in a step 502. Also, although a step 502 is depicted in Figure 5a, a step 502 could be used in other Figures and embodiments herein, include embodiments depicted in Figure 3a and Figure 5b. In addition, for a step 502, DPP server 103 can use an initiator bootstrap keys sequence number 197 in order to select the appropriate initiator bootstrap public key Bi 104a. For example, if a pair of initiator bootstrap keys had been deprecated (such as depicted in Figure 6a below), then an updated pair of initiator bootstrap keys should be selected using a device table 104d and sequence number 197 (also selected with device ID 206).

DPP server 103 can then send initiator 102' a message 503, where message 503 can include an identity 206 for device 101 (since initiator 102' may communicate with a plurality of responders 101x over time), the selected responder bootstrap public key Br 101a from a step 502, the selected initiator bootstrap public key Bi 104a, and optionally a value for the MAC address of the responder 101x comprising MAC.R 101d. Message 503 can be sent via the secure session 302b setup during a step 301 depicted in Figure 3a above.

Initiator 102' can receive message 503 and process the data received in a step 504. A step 504 can include validating the received public keys from a message 503 using public key validation steps from FIPS publication SP 800-56A (revision 2) and subsequent and related versions. Initiator 102' can use a step 504 in order to verify the public PKI keys received in a message 503 are consistent or comply with a selected set of cryptographic parameters 199a. At step 505 the initiator 102' can determine the initiator capabilities for conducting a DPP 191 with device 101 and responder 101x. In the exemplary embodiment depicted in Figure 5a, initiator 102' can select and use the capabilities of the role as a configurator. In other embodiments, an initiator 102' could select and use the capabilities of the role an enrollee for a step 504. In addition, an initiator 102' can generate a random number for an initiator nonce I-nonce 317a in a step 504.

Initiator 102' can then conduct a step 313 to derive the initiator ephemeral public key Pi 102a and the initiator ephemeral private key pi 102b. A step 313 was previously depicted and described above in Figure 3a. Although DPP server 103 conducted the step 313 in Figure 3a, an initiator 102' can also conduct the step 313 in system 500 in order to derive the initiator ephemeral PKI key pairs. Further, subsequent steps depicted in Figure 5a that were previously depicted as conducted by a DPP server 103 can be conducted by an initiator 102' using the embodiment depicted in Figure 5a. For example, depicted subsequent steps such as steps 314, 315, 319a, 320b, 322b, 328, etc. can be conducted, performed, or processed by initiator 102' using a DPP application 102g in an exemplary system 500, which could also represent a different mode 306a than system 300 in Figure 3a.

Initiator 102' in step 314 can use (i) the derived initiator ephemeral private key pi 102b from step 313, and (ii) the received responder bootstrap public key Br 101a from a message 503 to conduct the first initiator key exchange step 314, which is also depicted in Figure 4a above. The output or results from a step 314 can comprise a mutually derived shared secret key k1 with responder 101x. An intermediate step within the key exchange step 314 can include calculating the value M 410 (which could also be a secret shared key). Initiator 102' in step 315 can then conduct the encryption step 315 depicted in Figure 4a, and also described in Figure 3a. The plaintext input into a symmetric encryption algorithm 405a can comprise the initiator nonce 317a and initiator capabilities from a step 504 above. The shared secret key k1 314a can be used with symmetric

ciphering algorithm 405a in a step 315 from Figure 4a. The ciphertext output from an encryption step 315 can comprise ciphertext 315a.

At step 316, initiator 102' can calculate secure hash values for initiator bootstrap public key Bi 104a and responder bootstrap public key Br 101a. The selected hash 199e could be received by initiator 102' in a set of selected parameters 199a from a message 311 in a step 301. In exemplary embodiments, initiator 102' records the full set of cryptographic parameters 199, and a message to select a set of cryptographic parameters 199a could then specify a hash 199e to use with the public keys. For example, a selected set of cryptographic parameters 199a from message 311 in a step 301 could be the value "A", which would then be used by initiator 102' to determine that hash 199e in a set of cryptographic parameters 199 would be "SHA-256", as depicted in Figure 2b. Or, a message 311 in a step 301 could specify the full set of selected cryptographic parameters 199, such as listing out the individual values of key length 199b, curve name 199c, hash 199e, etc. The hash values calculated by initiator 102' in a step 316 in Figure 5a are depicted in Figure 5a as "H(Bi)" and "H(Br)", respectively.

Initiator 102' can then conduct a step 316a to send responder 101x a message 122, where message 122 can include the hash values for (i) the initiator bootstrap public key Bi 104a and (ii) the responder bootstrap public key Br 101a, the derived initiator ephemeral public key Pi 102a, and ciphertext 315a, where ciphertext 315a is encrypted with key k1 and includes an initiator nonce 317a and the initiator capabilities. Sending DPP authentication request message 122 was also depicted and described in connection with Figure 1b, Figure 1c, and Figure 3a above.

In Figure 5a, responder 101x can receive message 122 and conduct the series of steps 325. The series of steps 325 for responder 101x were depicted and described in connection with Figure 3a above, and steps 325 can comprise the steps 318, 319, 320, 314b, 321, 322, and 315b. Responder 101x can process the message 122 using steps 325 and generate a DPP authentication response 123. The steps 325 for a responder 101x to process the received message 122 and generate the response message 123 can comprise section 6.2.2 of DPP specification version 1.0. Other possibilities exist as well without departing from the scope of the present invention. In general, responder 101x can use the recorded bootstrap private keys and the derived ephemeral private keys in order to conduct ECDH key exchanges. Responder 101x can also (i) derive a responder nonce 318b, (ii) decrypt ciphertext in message 122 and (iii) encrypt ciphertext in message 123 using derived secret shared keys from the ECDH key exchanges. Responder 101x can send initiator 102' the DPP authentication response 123 after conducting the steps 325.

Initiator 102' can receive the DPP authentication response 123 and process the message. Using broadcast messages over connection 192, initiator 102' can identify the message from responder 101x based on the hash values for the bootstrap public keys in response 123. For embodiments where message 122 and response 123 are sent as unicast messages, initiator 102' can

identify the message from responder 101x based on the MAC.R 101d in the response 123. Initiator 102' can then extract and record responder ephemeral public key Pr 101e (depicted for a response 123 in Figure 3a). Initiator 102' can also a key verification step on the received key Pr 101e according to FIPS publication SP 800-56A (revision 2).

5 Initiator 102' can then send DPP server 103 a message 506 using secure connection 302b via access network 112, where message 506 can include (i) an identity for device 101 such as ID-token.device 206 (or ID.device 101i), and (ii) the responder ephemeral public key Pr 101e. In an exemplary embodiment a message 506 can include the MAC address MAC.R 101d to also identify device 101 and the associated key Pr 101e. DPP server 103 can receive the message 506 and record
10 the responder ephemeral public key Pr 101e along with the device identity in a server database 103a, as depicted in Figure 3b above.

DPP server 103 can then conduct an ECDH key exchange step 401c as depicted in Figure 4c in order to determine a secret shared key L 412. As depicted in Figure 5a and also Figure 4c, a step 401c can comprise DPP server 103 inputting (i) the recorded initiator bootstrap private key bi 104b
15 and (ii) a combination of the received responder ephemeral public key Pr 101e and the recorded responder bootstrap public key Br 101a into key exchange algorithm 401c in order to (iii) calculate the secret shared key L 412. Secret shared key L 412 can also comprise a secret shared key L according to section 6.2.4 of DPP specification version 1.0.

As depicted in Figure 5a, an initiator 102' can then conduct a second key exchange step
20 319a, where the second key exchange step 319a was depicted and described above in Figure 4b and also Figure 3a. Although a DPP server 103 was depicted as conducting a key exchange step 319a in Figure 3a, for the embodiments depicted in Figure 5a, an initiator 102' can conduct the key exchange step 319a. At step 319a, initiator 102' can input the received, validated responder ephemeral public key Pr 101e and the derived initiator ephemeral private key pi 102b into an ECDH key exchange
25 algorithm 401b in order to calculate a shared secret key N 411. Shared secret key N 411 can also be used by initiator 102' in a key derivation function 402c step below in Figure 5a. For a key exchange step 319a in Figure 5a, initiator 102' can input shared secret key N 411 into a key derivation function 402b in order to mutually derived the symmetric encryption key k2 314c. Note that secret shared key N 411 can also be used by initiator 102' for later use in a key derivation function step
30 402c, as depicted in Figure 5a.

As depicted in Figure 5a, initiator 102' can then conduct a decryption step 320b on received ciphertext 315c using the derived symmetric encryption key k2 314c from a step 319a above by initiator 102' in Figure 5a. The use and operation of a decryption step 320b was also depicted and described above in connection with Figure 4b and Figure 3a. Initiator 102' can read the plaintext
35 403b values from decryption step 320b. The plaintext values 403b can include the initiator nonce 317a from a message 122 above. The plaintext values 403b can also include a second ciphertext

322a (e.g. ciphertext 322a can be inside ciphertext 315c). Initiator 102' verifies that the initiator nonce 317a received in message 123 and decrypted in step 320b is equal to the initiator nonce 317a value sent in message 122 above. If the two initiator nonce 317a values are not equal, then initiator 102' can abort the DPP procedure and subsequent steps.

5 Initiator 102' can then receive a message 510 from DPP server 103, where message 510 can include an identity for device 101 such as ID-token.device 206 and the value L 412, where the value or secret shared key L 412 was calculated by DPP server 103 in a step 401c as depicted in Figure 5a. The value L 412 can be calculated by DPP server 103 using the initiator bootstrap private key bi 104b, and thus initiator 102' does not need to record or receive the initiator bootstrap private key bi
10 104b for embodiments depicted in Figure 5a.

Initiator 102' can then conduct a key derivation function step 402c using (i) the received value L 412 from message 510, (ii) the calculated shared secret key M 410 from a step 314, and (iii) the calculated shared secret key N 411 from a step 319a. A key derivation function step 402c is depicted and described in connection with Figure 4c above. The output from a key derivation
15 function step 402c conducted by initiator 102' can be a shared secret key ke 327a, as depicted in Figure 4c.

At step 328, initiator 102' can conduct a decryption step 328 on the ciphertext 322a using the key ke 327a from the previous step 402c. A decryption step 328 is depicted and described in connection with Figure 4c above. Initiator 102' can use the decryption step 328 in order to read the
20 plaintext value of R-auth 404 from the responder 101x in message 123. R-auth can comprise the value R-auth 404 from Figure 3a and Figure 4c and also comprise a hash 199e value that includes the combination of at least the initiator nonce 317a and the responder nonce 318b. At step 328, initiator 102' can then also internally calculate the same value for R-auth 404, and if the internally calculated value for R-auth 404 and the received, decrypted value for R-auth 404 are equal, then the
25 responder 101x can be considered authenticated. The calculation of R-auth 404 by initiator 102' in a step 328 can be according to section 6.2.3 of DPP specification version 1.0.

Initiator 102' can then conduct a step 322b, where step 322b can comprise calculating an initiator authentication value I-auth 404a. Initiator authentication value 404a can be similar to responder authentication value R-auth 404, except the values of initiator nonce 317a can be
30 sequenced before a responder nonce 318b inside a hash 199e calculation over the two nonces. Initiator 102' can then encrypt the initiator authentication value 404a using an encryption step 322 using key ke 327a, as depicted in Figure 4c above. In other words, an encryption step 322 in Figure 4c depicts the encryption of R-auth 404, and encryption step 322b can comprise initiator 102' encrypting the initiator authentication value I-auth 404a using the key ke 327a derived in a step
35 402c.

Initiator 102' can previously record the hash 199e values for the initiator and responder bootstrap public keys. Initiator 102 can take the recorded hash 199e values for the initiator and responder bootstrap public keys and append to them the encrypted value for I-auth 404a calculated by initiator 102 in step 322b. Initiator 102' can send responder 101x a DPP authentication confirm message 123a, as depicted in Figure 5a. The data and steps to determine and calculate a DPP authentication confirm message 123a can be according to section 6.2.4 of DPP specification version 1.0. As depicted in Figure 5a, the collection of steps and messages after a message 506 through transmitting a message 123a can collectively comprise a step 512.

Note that although the DPP authentication confirm message 123a can be equivalent to the DPP authentication confirm message from section 6.2.4 of DPP specification version 1.0, an important difference can be the operation of initiator 102' in Figure 5a versus an initiator as contemplated by DPP specification version 1.0. In the present invention as depicted in Figure 5a, initiator 102' does not need to record the initiator bootstrap private key bi 104b. So, using the technology depicted in Figure 5a, an initiator 102' can (i) send and receive messages that are fully compatible with a DPP protocol 191, but (ii) DPP server 103 can record key bi 104b. In other words, a responder 101x may not "know" that an initiator operates as either (i) an initiator 102* from Figure 1b or Figure 1c, or an initiator 102' from Figure 5a since the messages transmitted and received by the responder 101x can be fully compatible with a DPP 191.

The technology depicted in Figure 5a can enhance security since an initiator bootstrap private key bi 104b does not need to be shared with an initiator 102'. The initiator 102' can also not feasibly be able to determine the initiator bootstrap private key bi 104b using value L 412. By only recording the initiator bootstrap private key bi 104b in DPP server 103 (or also a device database 104), then a common, shared initiator bootstrap public key Bi 104a can be more securely shared across multiple devices 101, as contemplated by key table 104k in Figure 2b (with the depiction of "shared keys"). In other words, a manufacturer of device 101 could record a common initiator bootstrap public key Bi 104a across a plurality of devices 101, since a network 107 could share the initiator bootstrap public key Bi 104a with a device manufacturer of devices 101 (or a device distributor) before the devices are deployed near network access points 105. In other embodiments besides those depicted in Figure 5a, the initiator bootstrap private key bi 104b can be shared with initiator 102'.

In Figure 5a, responder 101x can receive message 123a and conduct the series of steps 339. The series of steps 339 for responder 101x were depicted and described in connection with Figure 3a above, and steps 339 can comprise the steps 328b, 331, 332, and 322c. Responder 101x can process the message 123a using steps 339 and generate a DPP configuration request 123b. In summary, responder 101x can decrypt the ciphertext in message 123a, where the ciphertext can include the initiator authentication value 404a. Responder 101x can also internally calculate a value for the

initiator authentication value 404a and the received value 404a, and if the two values are equal then responder 101x can consider the initiator 102' to be authorized. The responder 101x can take the roll of an enrollee and process a configuration attribute 330. The responder 101x can generate a random number comprising an enrollee nonce E-nonce 332a. Responder 101x can then conduct an encryption step 322c, where encryption step 322c can comprise (i) an encryption step 322 using key ke 327a, and (ii) the plaintext to be encrypted can be the E-nonce 332a and the configuration attribute 330. The output of an encryption step 322c can be a ciphertext 322d. Responder 101x can then send initiator 102' a DPP Configuration Request message 123b, where the configuration request 123b can include the ciphertext 322d, where the plaintext in ciphertext 322d can include E-nonce 332a and the configuration object 330. DPP Configuration Request message 123b can be processed by responder 101x according to section 6.3.3 of DPP specification version 1.0.

Initiator 102' can receive DPP Configuration Request message 123b via connection 192 and using DPP application 102g. As contemplated herein, a DPP application 102g can perform the steps for initiator 102' depicted in Figure 5a, such as transmitting/receiving DPP 191 messages using a first MAC.I 102n for connection 192 and transmitting/receiving messages with DPP server 103 using MAC.N 102q via access network 112. A DPP application 102g in initiator 102' can also perform the depicted steps for an initiator 102' in Figure 5a such as, but not limited to, 504, 505, 313, 314, etc. Initiator 102' can receive message 123b and process the message. Initiator 102' can conduct a decryption step 328c using decryption step 328 depicted in Figure 4c above. A difference between decryption step 328 and decryption step 328c is that decryption step 328c uses the ciphertext 332d as input into the symmetric decryption algorithm 405b. Initiator 102' can use the previously calculated key ke 327a from a step 402c in Figure 5a in order to read the plaintext within ciphertext 332d. Initiator 102' can record the plaintext E-nonce 332a and configuration attribute 330.

Initiator 102 can send DPP server 103 a message 513 via secure connection 302b from a step 301, where message 513 can include an identity for device 101 such as ID-token.device 206 and the plaintext value for the configuration attribute 330. Message 513 can be sent via secure session 302b, so the "plaintext" would be encrypted when transmitted over secure session 302b. In step 335, DPP server 103 can process the received configuration attribute 330. For the exemplary embodiment depicted in Figure 1a and Figure 5a, configuration attribute 330 can specify that device 101 with responder 101x may have a role of WiFi client. In exemplary embodiments, DPP server 103 can previously select a primary access network 308aa in a step 308a within a step 301 depicted in Figure 3a above, using the received networks available list 304c from initiator 102'. Also, in a step 308b in a step 301 in Figure 3a above, DPP server 103 could receive or record the set of network credentials 109 associated with the selected primary access network 308aa for device 101 using device identity ID-token.device 206 (or ID.device 101i).

In step 335, DPP server 103 can process the set of network credentials 109 for primary access network 308aa into a configuration object. The configuration object with network credentials 109 can comprise a connector or configuration object as specified in section 6.3.5 and 6.3.6 of DPP specification version 1.0. The configuration object with network credentials 109 can be optionally signed by DPP server 103 using the recorded initiator bootstrap private key bi 104b. In a different embodiment, a DPP server 103 can omit a signature for network credentials 109 in a step 335, and an initiator 102' can sign the network credentials 109 using the derived initiator ephemeral private key pi 102b. The steps to create and verify a signature with ECC PKI keys can be according to FIPS publication 186-4, and other possibilities exist as well. In Figure 5a, a DPP server 103 can also take the other steps described for a step 335 in Figure 3a above.

In a step 336, DPP server 103 can optionally encrypt network credentials 109 using an asymmetric encryption algorithm and a public key from the responder comprising one of (i) the responder bootstrap public key Br 101a or (ii) the derived responder ephemeral public key Pr 101e (received in message 506), or (iii) the public key for a device recorded by a certificate for device 101 in cert.device 101m. In some preferred exemplary embodiments, a step 336 can comprise encrypting network credentials 109 into credentials 109' using (i) the public key in cert.device 101m, and the elliptic curve Elgamal asymmetric encryption algorithm. The encryption algorithm could be from using Elgamal asymmetric encryption as summarized in the Wikipedia article for ElGamal encryption from July 10, 2018, which is herein incorporated by reference. A set of network credentials 109 that may be optionally encrypted using Elgamal asymmetric encryption and a responder public key is depicted as credentials 109' in Figure 5a. The optional Elgamal asymmetric encryption in a step 336 can use a set of parameters 199a in order to specify values such as encoding rules, byte or bit order, padding, etc. An optional encryption step 336 could also comprise (i) deriving a symmetric key and encrypting the symmetric key with Elgamal encryption and a responder public key and then (ii) encrypting the set of credentials 109 using a symmetric ciphering algorithm 405a and the derived symmetric key, in order to determine an encrypted set of credentials 109'.

In exemplary embodiments, an encrypted set of credentials 109' can be sent to initiator 102' instead of plaintext credentials 109, since an initiator 102' can read plaintext credentials 109, but not feasibly read encrypted credentials 109'. Note that encrypted credentials 109' can include a plaintext public key "h" or "g^x" along with the credentials 109' in order for responder 101x to use a private key such as responder bootstrap private key br 101b or responder ephemeral private key pr 101f to decrypt the encrypted credentials 109'. A reason that a step 336 can be performed in Figure 5a is that (i) initiator 102' could otherwise read and operate on a plaintext network credentials 109 (since initiator 102' uses encryption key ke 327a to encrypt plaintext 403c), and (ii) a DPP server 103 or a network 107 may prefer for initiator 102' to not read plaintext network credentials 109

(since initiator 102' may not be secured or under the full control of a network 107). In other embodiments, an encryption step 336 can be optionally omitted and plaintext credentials 109 can be sent to an initiator 102' via the secure session 302b via message 514.

In another embodiment, the asymmetric encryption step 336 can be performed by an external network, such that the external network generates the asymmetrically encrypted credentials 109', and the asymmetrically encrypted credentials 109' could be stored in a device credentials table 340 from Figure 3b. For example, credentials 109' could comprise the exemplary depicted credentials in table 340 of "from ...", where "..." was an external network in a networks available list 304c. In other words, a DPP server 103 may not be able to read the plaintext credentials 109 within an asymmetrically encrypted credentials 109', since an external network could encrypt the credentials using a step 336 before providing the credentials to network 107 and DPP server 103.

DPP server 103 can then send initiator 102' a message 514, where the message 514 can include an identity for device 101 such as ID-token.device 206 and the network credentials 109', where the network credentials 109' can be in the form of a configuration object and optionally encrypted using an Elgamal asymmetric encryption algorithm. Although a single value for network credentials 109' in message 514 are depicted, network credentials 109' could provide credentials for device 101 to connect with both a selected primary network 308aa and a selected backup network 308ab (with different credentials for each). Or, two sets of configuration objects with two sets of network credentials 109' could be sent in a message 514. Other possibilities exist as well without departing from the scope of the present invention.

Initiator 102' can receive message 514 and process the received credentials. At step 322e, initiator 102 can conduct an encryption step 322e using an encryption step 322 depicted and described in connection with Figure 4c above. Encryption step 322e can be performed using key ke 327a for device 101. For encryption step 322e, the plaintext value for the E-nonce 332a and the configuration object comprising network credentials 109' (where network credentials 109' may be optionally encrypted) can be input into a symmetric encryption algorithm 405a in order to generate a ciphertext 322f. Note that the use of asymmetrically encrypted credentials 109' can be optionally omitted, and an initiator 102' in a step 322e can input plaintext credentials 109 into a symmetric ciphering algorithm 405a.

Responder 101x can receive message 124 from initiator 102' with ciphertext 322f. Responder 101x can conduct a decryption step 328d using decryption step 328 depicted in Figure 4c below. A difference between decryption step 328 and decryption step 328d is that decryption step 328d uses the ciphertext 322f as input into the symmetric decryption algorithm 405b. Responder 101x can use the previously calculated key ke 327a in order to read the plaintext within ciphertext 322f. Note that the plaintext within ciphertext 322f can include an encrypted network credentials

109'. Responder 101x can confirm that plaintext in ciphertext 322f includes the previously sent E-nonce 332a.

For embodiments that use an optional encryption step 336 on network credentials 109 by DPP server 103 in step 336 (or an external network) in order to create encrypted network credentials 109', a responder 101x can use an asymmetric decryption step 337 in order to convert the encrypted network credentials 109' into plaintext network credentials 109. Responder 101x could use an Elgamal asymmetric decryption algorithm and either a recorded (i) responder bootstrap private key bi 101b or (ii) responder ephemeral private key pr 101f. Note that device 101 could decrypt credentials 109' using a private key for the public key in a certificate cert.device 101m, and in this case responder 101x could pass the received asymmetrically encrypted credentials 109' to device 101. Note that when using elliptic curve Elgamal asymmetric decryption, the responder 101x or device 101 uses the private key that corresponds to the public key used with asymmetric encryption of the credentials 109 in a step 336. Responder 101x could also use a selected set of cryptographic parameters 199a in order to conduct the Elgamal asymmetric decryption.

Note that encrypted network credentials 109' could include the public key "h" or "g" in order to conduct the Elgamal asymmetric decryption. For embodiments (a) without an optional encryption step 336 on network credentials 109 to create encrypted credentials 109', then (b) responder 101x can read the plaintext network credentials 109 in a step 337. Note that credentials 109 may be previously signed by DPP server 103 or initiator 102' using a private key for the initiator 102'. The responder could also verify a signature for the credentials 109 using a public key for the initiator, such as the initiator ephemeral public key Pi 102a or the initiator bootstrap public key Bi 104a.

After reading the plaintext network credentials 109 and optionally verifying a signature using a recorded public key 102a or 104a, device 101 can complete the configuration step 106 by loading the network credentials 109 and begin operating a WiFi radio 101h with the network credentials 109, as depicted in Figure 1a. In this manner, device 101 can obtain connectivity to network access point 105. In exemplary embodiments, the initiator 102' can conduct the initiator restore step 129 depicted and described in connection with Figure 1d above, such that initiator 102' can be returned to the previous state where WiFi radio 102w operates with a user configuration 131, thus minimizing impacts of the prior initiator configuration step 127 for an initiator user 102u. As depicted in Figure 5a, the collection of steps beginning with step 339 for responder 101x through step 106 for responder 101x can comprise a step 513.

Figure 5b

Figure 5b is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device database, a DPP server, an initiator proxy, and a responder, in accordance with exemplary embodiments. Before starting steps and message flows depicted in Figure 5b, the initiator, DPP server, and device database depicted for system 501 in Figure 5b may previously complete exemplary message flows and steps depicted for a step 301 in Figure 3a above. System 501 can include a device database 104, DPP server 103, initiator proxy 102', and responder 101x operating in device 101. Initiator proxy 102' can communicate with DPP server 103 via IP network 113 using an access network 112, also as depicted in Figure 3a above. As contemplated herein, an "initiator proxy 102'" depicted in Figure 5b may also be referred to as initiator 102', and an initiator 102' and initiator proxy 102' may be equivalent herein. A difference between system 501 in Figure 5b and system 300 in Figure 3a can be that the exemplary secret shared keys comprising key k2 314c and ke 327a are recorded and operated within an initiator 102'. In other words, a DPP server 103 does not need access to these keys for the embodiment depicted in Figure 5b. In exemplary embodiments depicted in Figure 5b, both (i) the initiator bootstrap private key bi 104b and (ii) the responder bootstrap public key Br 101a can remain recorded in a DPP server 103 and not exposed to initiator 102'. The exemplary embodiment depicted in Figure 5ba can comprise a third mode 306a of operation for initiator 102' and DPP server 103, as depicted in Figure 3b above.

After a step 301 from Figure 3a, DPP server 103 can conduct a step 502 in order to select a responder bootstrap public key Br 101a for device 101 using an identity ID-token.device 206. The value ID-token.device 206 could be received within step 301 in message 305 as depicted in Figure 3a. The responder bootstrap public key Br 101a could be recorded in a device database 104 or received from a device manufacturer for a device 101. Other possibilities exist as well for a network 107 to have access to a responder bootstrap public key 101a without departing from the scope of the invention. In some embodiments in order for a DPP server 103 to record a responder bootstrap public key Br 101a, an initiator 102' could conduct a PKEX key exchange in band via WiFi link 192 using PKEX key 198, where initiator 102' receives the key 198 in a message 311 in a step 301 in Figure 3a. Initiator 102' could (i) conduct the PKEX using key 198 with responder 101x during a step 303b, and then send the received responder bootstrap public key Br 101a to DPP server 103. Other embodiments depicted in Figure 3a and Figure 5a also could optionally use a PKEX with key 198 in order to receive the key Br 101a "in band" via WiFi link 192, such as if device database 104 does not record key Br 101a before a step 301 in Figure 3a. As mentioned for a step 502 in Figure 5a, a responder 101x could record a plurality of responder bootstrap public keys Br 101a, and the key 101a can be selected in a step 502. A MAC address for MAC.R 101d for device 101 using ID-token.device 206 can also be selected from a server database 103a in a step 502.

Initiator 102' can then conduct a step 313 to derive the initiator ephemeral public key Pi 102a and the initiator ephemeral private key pi 102b. A step 313 was previously depicted and described above in Figure 3a. Although DPP server 103 conducted the step 313 in Figure 3a, an initiator 102' can also conduct the step 313 in order to derive the ephemeral PKI key pairs. For example, initiator 102' also derives the initiator ephemeral PKI keys in a step 313 in Figure 5a. Initiator 102' can then send DPP server 103 the derived initiator ephemeral private key pi 102b in a message 521. Message 521 could be securely send to DPP server 103 via secure session 302b. Message 521 can also include device identity ID-token.device 206 (or ID.device 101i) in order for DPP server 103 to track the device 101 and initiator 102' associated with the key pi 102b. DPP server 103 can record the data received in message 521 in a server database 103a, as depicted in Figure 3b.

DPP server 103 for a step 314z can comprise DPP server 103 conducting a first initiator key exchange step 314 using an Elliptic Curve Diffie Hellman (ECDH) key exchange 401a, as depicted and described in connection with Figure 4a below. Data input for a step 314z can comprise the initiator ephemeral private key pi 102b and the recorded responder bootstrap public key Br 101a. Data output from a step 314z can comprise the secret shared key M 410 as described in Figure 4a above. In other words, the difference between a step 314z and a step 314 is the step 314z can terminate before the key derivation function 401a and the output of a step 314z can comprise the secret shared key M 410. In an alternative embodiment for system 501 in Figure 5b, DPP server 103 can also calculate key k1 using the full set of steps for a key exchange 314 (but key k1 may not be needed by DPP server 103 for the embodiment depicted in Figure 5b).

At step 316, DPP server 103 can calculate secure hash values for initiator bootstrap public key Bi 104a and responder bootstrap public key Br 101a, using a secure hash algorithm 199e. The values are depicted in Figure 3a as "H(Bi)" and "H(Br)", respectively. DPP server 103 can then send initiator 102' a message 522, where message 522 can include an identity for device 101 ID-token.device 206 (or ID.device 101i), the hash values for the initiator bootstrap public key Bi 104a and the responder bootstrap public key Br 101a, the shared secret key M 410 from a step 401a in key exchange step 314z, and the MAC address for the responder 101x comprising MAC.R 101d. Message 522 can be sent through secure session 302b and received and recorded by initiator 102'.

Initiator 102' can receive message 522 and conduct a series of steps in order to create and process a DPP authentication request message 123. Initiator 102' can input the received value M 410 from a message 522 into a key derivation function 402a in order to derive a secret shared key k1 314a, depicted as a step 402a for initiator 102' in Figure 5b. At step 504 the initiator 102' can determine the initiator capabilities for conducting a DPP 191 with device 101 and responder 101x. In the exemplary embodiment depicted in Figure 5b, initiator 102' can select and use the capabilities of the role as a configurator. In other embodiments, an initiator 102' could select and use the

capabilities of the role an enrollee for a step 504. In addition, an initiator 102' can generate a random number for an initiator nonce I-nonce 317a in a step 504. A step 504 can also take place before a step 402a in some exemplary embodiments.

Initiator 102' in step 315 can then conduct the encryption step 315 depicted in Figure 4a, and also described in Figure 3a and Figure 5a. The plaintext input into a symmetric encryption algorithm 405a can comprise the initiator nonce 317a and initiator capabilities from a step 504 above. The shared secret key k1 314a can be used with symmetric ciphering algorithm 405a in a step 315. The ciphertext output from an encryption step 315 can comprise ciphertext 315a.

Initiator 102' can then conduct a step 316a to send responder 101x a message 122, where message 122 can include the hash values for (i) the initiator bootstrap public key Bi 104a and (ii) the responder bootstrap public key Br 101a, the derived initiator ephemeral public key Pi 102a (from a step 313 above), and ciphertext 315a, where ciphertext 315a is encrypted with key k1 and includes an initiator nonce 317a and the initiator capabilities. Sending DPP authentication request message 122 was also depicted and described in connection with Figure 1b, Figure 1c, Figure 3a, and Figure 5a above.

In Figure 5b, responder 101x can receive message 122 and conduct the series of steps 325. The series of steps 325 for responder 101x were depicted and described in connection with Figure 3a above, and steps 325 can comprise the steps 318, 319, 320, 321, 314b, 322, and 315b. Responder 101x can process the message 122 using steps 325 and generate a DPP authentication response 123. The steps 325 for a responder 101x to process the received message 122 and generate the response message 123 can comprise section 6.2.2 of DPP specification version 1.0. Other possibilities exist as well without departing from the scope of the present invention. In general, responder 101x can use the recorded bootstrap private keys and the derived ephemeral private keys in order to conduct ECDH key exchanges. Responder 101x can also (i) derive a responder nonce 318b, (ii) decrypt ciphertext in message 122 and (iii) encrypt ciphertext in message 123 using derived secret shared keys from the ECDH key exchanges. Responder 101x can send initiator 102' the DPP authentication response 123 after conducting the steps 325.

Initiator 102' can receive the DPP authentication response 123 and process the message. Using broadcast messages over connection 192, initiator 102' can identify the message from responder 101x based on the hash values for the bootstrap public keys in response 123. For embodiments where message 122 and response 123 are sent as unicast messages, initiator 102' can identify the message from responder 101x based on the MAC.R 101d in the response 123. Initiator 102' can then extract and record responder ephemeral public key Pr 101e (depicted for a response 123 in Figure 3a). Initiator 102' can also a key verification step on the received key Pr 101e according to FIPS publication SP 800-56A (revision 2).

Initiator 102' can then send DPP server 103 a message 506 using secure connection 302b via access network 112, where message 506 can include (i) an identity for device 101 such as ID-token.device 206 (or ID.device 101i), and (ii) the responder ephemeral public key Pr 101e. In an exemplary embodiment a message 506 can include the MAC address MAC.R 101d to also identify device 101 and the associated key Pr 101e. DPP server 103 can receive the message 506 and record the responder ephemeral public key Pr 101e along with the device identity in a server database 103a, as depicted in Figure 3b above.

As depicted in Figure 5b, initiator 102', DPP server 103, and responder 101x can then conduct a step 512. A step 512 can comprise the series of steps and message depicted and described in connection with Figure 5a above. In summary, initiator 102' can conduct a key exchange step 319a to derive the key k2 314c and then decrypt the ciphertext from message 123 into plaintext using a step 320b. Key exchange step 319a can also calculate a shared secret key N 411 as depicted in Figure 4b. The plaintext from a step 320b can comprise a second ciphertext 322a encrypted with a key ke 327a. DPP server 103 can (i) conduct a key exchange step 401c in order to derive shared secret key L 412. DPP server 103 can send initiator 102' the shared secret key L 412. Initiator 102' can conduct a key derivation function step 402c from Figure 4c using the values M 410 (from message 522), N 411 (from a step 319a), and L 412 (from DPP server 103 in a message 510 in a step 512) in order to derive key ke 327a, as depicted in Figure 4c. Initiator 102' can decrypt the second ciphertext 322a in message 123 into plaintext with a decryption step 328 using the derived key ke 327a and read the value R-auth 404. Initiator 102' can also use a step 328 to compare the received, decrypted value for R-auth 404 with an internally calculated value for R-auth, and responder 101x can be authenticated if the two values for R-auth 404 are equal. Initiator 102' can calculate a value for I-auth 404a and encrypt the value with key ke 327a in a step 322b. Initiator 102' can then send responder 101x a DPP authentication confirm message 123a in a step 512, as depicted in Figure 5a and Figure 5b.

As depicted in Figure 5b, initiator 102', DPP server 103, and responder 101x can then conduct a step 513. A step 513 can comprise the series of steps and message depicted and described in connection with Figure 5a above. In summary, responder 101x can use a step 339 to receive and process the DPP authentication confirm message 123a. Responder 101x can decrypt the I-auth value 404a in a step 331. Responder 101x can calculate the value for I-auth 404, and if the received and calculated values for I-auth 404a are equal, then initiator 102' can be authenticated with responder 101x. Responder 101x can send a DPP configuration request 123b message, which can include a ciphertext 322d an E-nonce 332a and a configuration attribute 330. Initiator 102' can decrypt the ciphertext 322d and send DPP server 103 the configuration attribute 330. DPP server 103 can send initiator 102' a set of network credentials 109. The set of credentials 109 may optionally be encrypted with an asymmetric encryption algorithm using a public key for the responder, and for

these embodiments the set of credentials 109 can be designated an encrypted set of credentials 109'. The public key for the responder with an optional asymmetric encryption for the credentials 109' could be (i) the responder bootstrap public key Br 101a, or (ii) the responder ephemeral public key, or (iii) a private key for the device public key in cert.device 101m (e.g. step 337 in Figure 5a).
5 Credentials 109 or credentials 109' can also be signed by either (i) the DPP server 103 using the recorded initiator bootstrap private key bi 104b, or (ii) using the derived initiator ephemeral private key pi 102b.

Continuing with a step 513 in Figure 5b, the credentials 109 or credentials 109' could be signed by both the DPP server 103 and the initiator 102 in a step 513. The initiator 102' can receive
10 the signed network credentials 109' or 109 and encrypt the credentials and the E-nonce 332a using the derived key ke 327a. The initiator 102' can send responder 101x a DPP configuration response 124 message with the encrypted and signed credentials for device 101. Responder 101 can verify the signature and decrypt the credentials 109 or asymmetrically encrypted credentials 109'. Device 101 can apply the credentials 109 and connect with another node using the credentials 109, and the
15 other node could be network access point 105. The other node could also be a WiFi peer-to-peer (P2P) connection. In another embodiment, device 101 could operate as an access point 105 for other WiFi clients, and in this embodiment the credentials 109 received in a message 124 could be for device 101 to operate as a WiFi access point using the credentials 109. Other possibilities exist as well for a responder 101x to receive a set of credentials 109 from an initiator 102' without departing
20 from the scope of the present invention.

System 501 as depicted in Figure 5b can provide several benefits for conducting a device provisioning protocol, compared to the conventional technology depicted in Figure 1b and Figure 1c. First, an initiator bootstrap private key bi 104b can remain recorded in a network 107 and does not need to be shared with an initiator 102'. The operating environment and security for an initiator
25 102' or initiator 102* in Figure 1b and Figure 1c may be outside the control of a network 107 or a device 101 with responder 101x. If an initiator bootstrap private key bi 104b is sent to initiator 102' or initiator 102*, then the initiator bootstrap private key bi 104b cannot be securely reused (since initiator 102' or initiator 102* could then share the key bi 104b or attempt to reuse it with other devices 101 recording the same initiator bootstrap public key Bi 104a). Further, recording the same,
30 shared initiator bootstrap public key Bi 104a across a plurality of devices 101 (such as during device manufacturing or before distribution) can provide benefits for scalability and ease of deployment, especially if the initiators 102' or initiators 102* are not known to a network 107 before distribution of devices 101.

In other words, recording a same initiator bootstrap public key Bi 104a across a plurality of
35 devices may not be securely feasible if the initiator bootstrap private key bi 104b needs to be used by multiple, insecure initiators 102' or initiators 102*. In addition, the responder bootstrap public key

Br 101a can remain securely recorded in a network 107 and not exposed to an initiator 102' for a system 501 in Figure 5b (where Br 101a can be recorded in a device database 104 instead of being received by an initiator 102' from a message 503 in Figure 5a). During the lifetime of a device 101 that could be a decade or longer, a device owner or device user may prefer the device 101 to conduct a device provisioning protocol multiple times, such as after a "factory reset" for device 101. Keeping or storing the responder bootstrap public key Br 101a and initiator bootstrap private key bi 102b in a network 107 can support the steps for a device provisioning protocol to be securely reused for the same device 101.

Yet another benefit of a system 300, system 500, and system 501 is that an overall trust level by responder 101x for conducting a device provisioning protocol can be enhanced or increased, since both an initiator 102' and a DPP server 103 must work in conjunction in order to successfully complete a DPP 191. In other words, for a system 300, system 500, and system 501, security risks for an insecure initiator 102' can be compensated by likely a higher overall level of security for a DPP server 103. These are just a few of the exemplary benefits for systems 300, 500, and 501 compared to conventional technology for conducting a device provisioning protocol, and other benefits will be available as well for those of ordinary skill in the art.

Figure 6a

Figure 6a is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device database, a DPP server, a WiFi access point, and a device, in accordance with exemplary embodiments. Before starting steps and message flows depicted in Figure 6a, a device 101 with responder 101x could have completed the exemplary configuration step 106 as depicted in Figures 1a, 3a, 5a, and 5b above in order for device 101 to receive and load a set of network access credentials 109. System 600 can include a device database 104, a DPP server 103, a WiFi access point 105, and a device 101. WiFi access point 105 can broadcast an SSID with a value of SSID.network-AP 109a, which also could have been observed by an initiator 102' in a step 303a to collect a networks available list 304c. A DPP server 103 (or another server within a network 107) could (i) conduct a step 308a to select WiFi access point 105 from the networks available list 304c and then (ii) obtain the credentials 109 using a step 308b, and then send the credentials 109 through initiator 102' in a DPP message 124.

Configuration data for device 101 to utilize with WiFi access point 105 could be included in data and values for network credentials 109 for device 101. For embodiments where a selected access network 308aa for device 101 is not a WiFi access point 105 (such as possibly a 5G wireless WAN connection), then network credentials 109 could comprise a profile for an embedded universal integrated circuit card (eUICC) as specified in GSMA standards SGP.22 and SGP.02. For

embodiments where credentials 109 comprise an eUICC profile, then the credentials 109 may preferably be asymmetrically encrypted into credentials 109' using a step 336 from Figure 5a by an external network recording the eUICC profile, and then asymmetrically decrypted by responder 101x or device 101 using a step 337 from Figure 5a. DPP server 103 would record the eUICC profile as credentials 109' (or "from ..." in Figure 3b). For embodiments where asymmetrically encrypted credentials 109' comprising an eUICC profile are received by responder 101x in a DPP Configuration Response 124 message, then device 101 can also operate an eUICC as specified in as specified in GSMA standards SGP.22 and SGP.02.

Values or data in credentials 109 could comprise frequencies or channels to utilize, a service set identifier (SSID) or network name, user identities and passwords, and/or public certificates for device 101, public certificates for a certificate authority, etc. As contemplated herein, a set of credentials 109 can include supporting data to use the credentials 109 and/or connect with an access network 112 or network 107, such as configuration data for a WiFi radio 101h in device 101. The supporting data could comprise Config.network-AP 109c as depicted and described in connection with Figure 1a. Data collected for a set of credentials 109 was also described in a step 335 in Figure 3a.

Device 101 and access point 105 can conduct a WiFi session setup 602 using the WiFi radio 101i with credentials 109. WiFi connection setup 602 can utilize standard WiFi security protocols such as, but not limited to, WPA2, WPA3, or subsequent and related standards. In exemplary embodiments, WiFi connection setup 602 utilizes PMK and a PMKID, where PMK and PMKID can be uniquely associated with device 101. Or WiFi connection setup 602 could use a preshared secret key (PSK) in credentials 109. At step 603, device 101 can select the DPP server 103 and a URL for DPP server 103. A URL for DPP server 103 could be included with credentials 109, or otherwise fetched by device 101 from an IP network 113 after session setup 602. A step 603 can also comprise device 101 confirming expected connectivity to an IP network 113 is provided via access point 105, such as performing domain name server (DNS) requests and downloading data via HTTP or HTTPS/TLS from an IP network 113.

Device 101 and DPP server 103 can then conduct a secure session setup 604. Secure session setup 604 could comprise a session using TLS, IPSec, VPN, SSH, or similar secure networking technologies to establish secure sessions. In exemplary embodiments, DPP server 103 uses the responder bootstrap public key Br 101a as a public key for device 101 in secure session 603 and device 101 uses the initiator bootstrap public key Bi 104a as a public key for DPP server 103 in secure session 603. In other words, (i) DPP server 103 uses the initiator bootstrap private key bi 104b as the private key for DPP server 103 in secure session 603, and (ii) device 101 uses the responder bootstrap private key br 101b as the private key for device 101 in secure session 603. In exemplary embodiments, the public keys responder bootstrap public key Br 101a and the initiator

bootstrap public key Bi 104a are not transmitted between the nodes in a secure session setup 603, but rather the public keys are selected for use since they are already recorded by the two nodes. In this manner, security of secure session 603 can be enhanced, because the public keys are not transmitted or received in a plaintext form, but rather are already previously recorded by the two nodes. In an exemplary embodiment, secure session setup 603 could designate or identify the initiator bootstrap public key Bi 104a (and corresponding private key) for device 101, which could be selected by DPP server 103 sending an initiator bootstrap keys sequence number 197 in a secure session setup 603, for embodiments where device 101 can record a plurality of initiator bootstrap PKI keys, as depicted in device table 104d in Figure 2b. Note that a secure session setup 604 could also use different public keys than those depicted in Figure 6a, such as a certificate for DPP server 103 cert.DPPS 103c and a certificate for device 101 cert.device 101m.

At step 605a, device 101 can download and install a configuration package for device 101 using WiFi session 602 in Figure 6a. The configuration package can be downloaded from network 107, and also potentially from DPP server 103 using secure session 604. Configuration package for device 101 can comprise an updated operating configuration for device 101. Configuration package for device 101 could be downloaded from a network 107. Configuration package for device 101 could include updated software or firmware for device 101, different credentials for device 101 to use when connecting with servers within a network 107, such as potentially a reporting server or a monitoring server within a network 107, timers for device 101 to use with servers in a network 107, and other data as well for the ongoing operation of device 101 with a network 107. After internally recording or loading the files for a configuration package in a step 605a, device 101 can perform a reboot, so that device 101 restarts with the new files from the configuration package. Upon a reboot in a step 605a, connections 602 and 603 may temporarily terminate with the reboot, but device 101 can re-establish connections 602 and 603 after reboot. A step 605a can then also comprise device 101 creating a report 605aa, where report 605aa includes a status code with success or errors for each file in a configuration package. In other words, report 605aa can record the success or errors of applying each of the files in a configuration package, which may be useful for network 107 in system 100 to know the configured state of device 101 after a step 605a. Note that step 605a can take place before a secure session setup 604.

At step 605b device 101 can perform an RF frequency scan or sweep all available mobile network operators and network access technologies around device 101 in order to collect a second networks available list 304a. The second networks available list 304c can be useful for DPP server 103 or network 107 to confirm that a selected primary access network 308aa for device 101 is in fact a preferred network. An exemplary first networks available list 304c was depicted in Figure 3a above (for initiator 102'). Although initiator 102' also collected a first networks available list 304c, device 101 can operate with a different antenna and set of radios 101h with different capabilities

than initiator 102'. For example, RSSI or RF signal strength measurements by initiator 102' in a step 303a may be different than for a device 101. As another example, device 101 may operate at a different location (such as an exemplary tens of meters away) from the location of initiator 102' in a step 303a, and thus a preferred access network 112 for device 101 may be different using the second networks available list 304a from device 101 in step 605b than a first networks available list 304a from initiator 102'. In other words, a network selection step 308a may provide an initial estimate for a preferred access network 112 (identified by selected primary network 308aa in Figure 3b), but the actual/updated preferred access network 112 for device 101 using the criteria from a step 308a could be selected in a step 605b using the second networks available list 304c from device 101. A step 605b for device 101 can include the actions for an initiator 102' in a step 303a to collect a first networks available list 304a for initiator 102'.

Device 101 can send DPP server 103 a message 606, where message 606 can include a device identity ID-token.device 206, the second networks available list 304c collected by device 101 in a step 605b, an initiator bootstrap sequence number 197 for the most recent initiator bootstrap public key Bi 104a used by device 101, and a report 605aa, which could comprise the report generated in a step 605a. As depicted in Figure 6a, message 606 can be sent via (i) access point 105 and then (ii) access network 112 for access point 105. Although a DPP server 103 is depicted in Figure 6a as communicating with device 101 and receiving message 606, another server in a network 107 or associated with network 107 could receive the message 606 (using a secure session similar to secure session 604 depicted in Figure 6a). Note that message 606 could also include the most recent initiator bootstrap public key Bi 104a recorded by device 101.

A DPP server 103 could then conduct a step 308a using the second networks available list 304c from device 101, in order to select an updated, selected primary access network 308aa. A step 308a was depicted and described above in connection with Figure 3a. For embodiments where a step 308a by DPP server 103 selects a different, second primary access network 308aa for device 101 in Figure 6a (e.g. different than a first primary access network 308aa in Figure 3a), a DPP server 103 could then conduct a step 308b in order to receive an updated, second set of credentials 109 for the second primary access network 308aa. Note that the credentials 109 received and recorded in a step 308b by DPP server 103 can comprise an asymmetrically encrypted credentials 109', where a server functioning as the source of asymmetrically encrypted credentials 109' could perform a step 336 depicted and described in connection with Figure 5a. In exemplary embodiments, only device 101 could feasibly read the plaintext credentials 109 for an asymmetrically encrypted credentials 109', where device 101 would use a recorded private key (e.g. key 101b, 101f, or the private key for cert.device 101m) in order to perform an elliptic curve Elgamal asymmetric decryption on the credentials 109'.

A DPP server 103 could conduct a step 607, where DPP server 103 can process the received report 605aa in order to confirm a configuration package for device 101 was successfully applied in a step 605a. DPP server 103 could also forward report 605aa to other servers in a network 107 in a step 607. If errors or problems are detected in report 605aa in a step 607, then DPP server 103 can send a command to device 101 to back out the configuration package (e.g. uninstall) or restore a previous state for device 101.

At step 608, DPP server 103 can determine if recorded initiator bootstrap public key Bi 104a for device 101 should be updated. There could be many reasons both device 101 and network 107 may prefer to update initiator bootstrap public key Bi 104a for a responder 101x operating in device 101. A device 101 may have a lifetime of a decade or longer, and increased key security, use of different cryptographic algorithms, or different cryptographic parameters 199 may be preferred over time. Another reason could be that device 101 or network 107, after (a) a first configuration step 106 via a first session of a device provisioning protocol, may prefer (b) to support a second configuration step 106 via a second session of a device provisioning protocol, where the second session could use a different initiator bootstrap PKI key pair. Another reason could be that a first device 101 used the same key (e.g. same numeric value) for an initiator bootstrap public key Bi 104a as second device 101 (or possibly multiple other devices 101), which is also depicted for “shared keys” in a key table 104k in a device database 104 in Figure 2b. Security could be increased for the first device 101 by changing the recorded initiator bootstrap public key Bi 104a after the key has been used. Other reasons for a device 101, DPP server 103, or network 107 may exist for a DPP server 103 to determine key Bi 104a should be updated for device 101, without departing from the scope of the present invention. Figure 6 depicts an embodiment where DPP server 103 selects to update the initiator bootstrap public key Bi 104a for device 101 in a step 608.

At step 609, DPP server 103 can derive a new, updated initiator bootstrap public key Bi 104a and a corresponding new, updated initiator bootstrap private key bi 104b. The DPP server 103 can use a selected set of cryptographic parameters 199a to derive the keys. The DPP server 103 can use a random number generator and a key pair generation algorithm in order to derive the new, update initiator bootstrap PKI keys. The DPP server 103 can use the step 313 depicted and described in Figure 3a to derived the initiator bootstrap PKI keys (e.g. using same or similar steps as deriving initiator ephemeral PKI keys). Although Figure 6a depicts DPP server 103 as conducting a step 609 to derive new initiator bootstrap PKI keys, another server in a network 107 could conduct a step 609, including device database 104. At step 609, DPP server 103 can also update initiator bootstrap keys sequence number 197 recorded in a device database 104 depicted in Figure 2b to reflect or account for the use of new initiator bootstrap PKI keys. The updated number 197 is depicted in Figure 6a an initiator bootstrap keys sequence number 197a. In exemplary embodiments, the initiator bootstrap keys sequence number 197 can be used at a later time by DPP server 103 when conducting a step

302d for device 101 (e.g. querying device database 104 for initiator bootstrap PKI keys in Figure 3a), such as if DPP server 103 supports a device provisioning protocol for device 101 again at a later time.

A DPP server 103 can then send device database 104 a message 609a, where message 609a can include a device identity ID-token.device 206 (or ID.device 101i), the new, updated initiator bootstrap public key Bi 104a and the corresponding new, updated initiator bootstrap private key bi 104b, where the PKI keys were generated in a step 609. Message 609a can also include the updated initiator bootstrap keys sequence number 197a. Note that DPP server 103 and device database 104 can continue to use secure session 302a from a Figure 3a in order to secure message 609a. In some embodiments, initiator bootstrap private key bi 104b can be recorded only within DPP server 103 in a server database 103a, and for these embodiments initiator bootstrap private key bi 104b can be omitted from a message 609a.

At step 610, device database 104 can record the data from a message 609a in a device table 104d for device 101 using ID-token.device 206 (or ID.device 101i). The data can be recorded by adding a row in device table 104d for device 101. The new, updated initiator bootstrap public key Bi 104a and the corresponding new, updated initiator bootstrap private key bi 104b could be recorded in the added row, along with the received value for an initiator bootstrap keys sequence number 197a. In this manner, device database 104 can record and identify the initiator bootstrap PKI keys for use with device 101 at a later time, for reasons such as, but not limited to, the reasons discussed in connection with step 608 above. Database 104 can then send DPP server 103 a message 611, where message 611 can confirm that step 610 was successfully completed and the new initiator bootstrap PKI keys were recorded in a device table 104d.

After receiving message 611, DPP server 103 can send device 101 a message 612 via secure session 604. Message 612 can include the new, updated initiator bootstrap public key Bi 104a, which was previously derived by DPP server 103 in a step 609 and recorded by device database 104 in a step 610. Message 612 can also include (i) a selected set of parameters 199a associated with the new Bi 104a (e.g. key length, ECC curve name, etc.), and (ii) the updated initiator bootstrap keys sequence number 197a. Device 101 can receive message 612 with the new, updated initiator bootstrap public key Bi 104a and updated initiator bootstrap keys sequence number 197a.

At step 613, device 101 could record the data received in message 612 with a responder 101x operating in device 101. In exemplary embodiments, in step 612, device 101 could also use the received parameters 199a to derive an updated responder bootstrap private key br 101b and responder bootstrap public key Br 101a. Device 101 could use the equivalent steps as DPP server 103 in step 313 depicted and described in Figure 3a in order for responder 101x to derive a new, updated bootstrap private key br 101b and responder bootstrap public key Br 101a. A step 613 can also comprise device 101 recording the derived PKI keys and received key Bi 104a in nonvolatile

memory of device 101. Device 101 can also update a responder 101x operating in device 101 to use the new PKI keys for any subsequent, later instance of a device provisioning protocol either with an initiator 102' or an initiator 102*.

5 In message 614, device 101 can send DPP server 103 and "OK" message noting the successful receipt and recording of the new, updated initiator bootstrap public key Bi 104a. Message 614 can also include the new, updated responder bootstrap public key Br 101a. At a step 615, DPP server 103 can receive message 614 and record the data in a server database 103a for a device 101 using device identity ID-token.device 206. In exemplary embodiments, the new, updated responder bootstrap public key Br 101a can be used by DPP server 103 when supporting an initiator 102' to
10 conduct a device provisioning protocol with device 101 and responder 101x at a later time.

DPP server 103 can then send device database 104 a message 615, where message 615 can include a device identity ID-token.device 206 (or ID.device 101i) and a confirmation that device 101 received and recorded the new, updated initiator bootstrap public key Bi 104a. In this manner, device database 104 can confirm that the new key Bi 104a with corresponding private key bi 104b is
15 "ready for use", and can be selected in a subsequent query 309 (in Figure 3a) from a DPP server 103 for device 101 with identity ID-token.device 206. As depicted in Figure 6a, message 615 can also include a new, updated responder bootstrap public key Br 101a. Device database 104 can also record the received new key Br 101a in a key table 104k for device 101 with identity ID-token.device 206 as depicted in Figure 2b. In this manner, device database 104 can select the new,
20 updated responder bootstrap public key Br 101a (along with initiator PKI keys) in a subsequent query 309 from a DPP server 103 for device 101, in order to conduct a later device provisioning protocol 191.

Initiator 102' can conduct a step 617, where step 617 can comprise the DPP application 102g or initiator 102' sending a query 618 to DPP server 103 in order to confirm successful completing of
25 configuration step 106. DPP server 103 could determine if a configuration step 106 is complete based on the successful establishment of secure session 604 and receipt of a report 605aa. DPP server 103 could also determine that a configuration step 106 is complete using a report 605aa and any of: (i) device 101 has wireless connectivity properly established for a selected primary network 308aa and/or selected backup network 308ab (e.g. credentials 109 were received by device 101 and
30 installed), (ii) device 101 has successfully collected a networks available list 304c, (iii) a configuration package was downloaded and installed in a step 605, (iv) device 101 has received an updated initiator bootstrap public key Bi 104a, and/or (v) device 101 has successfully updated responder bootstrap PKI keys.

As depicted in Figure 6a, DPP server 103 can reply to query 618 with a message 619 of
35 "device OK", if configuration step 106 has been confirmed by DPP server 103 as completed, such as through any of (i) through (v) in the paragraph above. Individual components within (i) through (v)

in the paragraph above could be recorded or confirmed in a report 605aa. Not all of (i) through (v) are required for some embodiments, although connectivity for device 101 to an IP network 113 can be required for DPP server 103 to consider a configuration step 106 completed. In exemplary embodiments, upon successful receipt of a message 619 with a status of "device OK" for device 101 or an equivalent message 619, then initiator 102' can display to initiator user 102u that a configuration step 106 for device 101 has been completed and device 101 can be left in operation, potentially without additional steps or manual changes to be performed by initiator user 102u. For embodiments where message 619 provides a status of "device not OK" or equivalent, then initiator 102' could display to initiator user 102u an error status and initiator user 102u could work with device 101 and network 107 in order to diagnose and rectify an error code or error condition in order to receive a subsequent message 619 of "device OK".

Figure 6b

Figure 6b is a graphical illustration of an exemplary system, where a responder records a plurality of initiator bootstrap public keys and the responder selects and uses an initiator bootstrap public key, in accordance with exemplary embodiments. The present invention contemplates that the systems and methods for (i) using an initiator 102' with a DPP server 103 and a responder 101x for a device provisioning protocol, where (ii) a responder 101x can record a plurality of initiator bootstrap public keys Bi 104a. Benefits can include the plurality of initiator bootstrap public keys Bi 104a can be recorded in a device 101 with responder 101x before device 101 is taken to the physical location of a network access point 105. The plurality of initiator bootstrap public keys can be written to device 101 during manufacturing or distribution, thereby increasing the flexibility of (a) recording at least one compatible or supported initiator bootstrap public key Bi 104a in a device 101 for initiator 102', before (b) an initiator 102' is selected or known for conducting the device provisioning protocol steps 191 with a responder 101x. In exemplary embodiments, different values for recorded initiator bootstrap public keys Bi 104a can represent different possible networks 107 for configuration 106 of device 101 with responder 101x. As contemplated herein, (a) pre-recording a plurality of initiator bootstrap public keys Bi 104a in a plurality of devices 101 can still keep the various systems depicted herein secure, because (b) the corresponding initiator bootstrap private keys bi 104b can remain recorded individually and separately within each of a plurality of networks 107 which can conduct the device provisioning protocol 191 with the plurality of devices 101 recording the same values for initiator bootstrap public keys Bi 104a.

System 601 in Figure 6b can include an initiator 102' and a device 101, where device 101 can operate a responder 101x. Initiator 102' in Figure 6b can operate as an initiator 102' in Figure 1a, Figure 3a, Figure 5a, etc. above. Responder 101x can record (i) responder bootstrap PKI keys Br

101a and br 101b, as well as (ii) at least one of a set of selected cryptographic parameters 199a. Although Figure 6b depicts only a single set of PKI keys Br 101a and br 101b, and a set of selected cryptographic parameters 199a, a responder 101x could also record a plurality of different sets of PKI keys Br 101a and br 101b, with different parameters 199a (e.g. a first set of responder bootstrap keys for a first ECC curve P-256, a second set of bootstrap keys for a second ECC curve P-384, etc.). Responder 101x could determine which of the different sets of responder bootstrap PKI keys to utilize for conducting a device provisioning protocol 191 based on the received hash value for a responder bootstrap public key Br 101a received in a message 122. Although not depicted in Figure 6b, a responder 101x could pre-compute the hash values for each of a possible plurality of bootstrap public keys Br 101a using a hash 199e using cryptographic parameters 199a for each different responder bootstrap public key pair. Responder 101x could (i) listen for receiving a hash 199e value for any of the recorded bootstrap public keys Br 101a in an incoming message 122. When responder 101x observes a matching hash 199e value for a recorded key Br 101a, then select the private key br 101b for use in creating a response message 123. In other embodiments, a single recorded set of bootstrap PKI keys Br 101a and br 101b can be sufficient in order for device 101 to securely conduct a device provisioning protocol 191.

In some embodiments of the present invention, a responder 101x could record a plurality of different initiator bootstrap public keys Bi 104a in an initiator bootstrap public keys table 622, as depicted in Figure 6b. Each of the different initiator bootstrap public keys 104a could also be recorded with both (i) a hash 199e value for each individual key Bi 104a, and (ii) a selected set of cryptographic parameters 199a. Each of the different initiator bootstrap public keys 104a could be associated with a different network 107, as depicted by the exemplary values for possible networks 107 (e.g. "Google", "Amazon", etc.). The corresponding bootstrap private key bi 104b for each of the different depicted initiator bootstrap public keys Bi 104a could be recorded in each of the different networks 107. The ninth listed exemplary network 107 in a table 622 in responder 101x can comprise the exemplary network 107 depicted in Figure 1a, which can also operate the DPP server 103 in Figure 3a or Figure 5a, etc.

Although a single initiator bootstrap public key Bi 104a is depicted in Figure 6b for each different exemplary network 107, each different exemplary network 107 could also record multiple different initiator bootstrap public keys Bi 104a, such as (i) a first initiator bootstrap public key Bi 104a for a network 107 with a first set of selected cryptographic parameters 199a (e.g. with exemplary curve P-256), and (ii) a second initiator bootstrap public key Bi 104a with a second set of selected cryptographic parameters 199a (e.g. with exemplary curve P-384). Other possibilities exist as well for recording a plurality of keys Bi 104a for the same network 107 in a table 622 without departing from the scope of the present invention.

Further, the exemplary recorded initiator bootstrap public key Bi 104a for multiple networks 107 in a table 622 can either comprise (i) a “shared key” depicted and described in a key table 104k in Figure 2b where the value or number for the key Bi 104a can be recorded across a plurality of different devices 101, or (ii) a “unique key”, where the value or number for the key Bi 104a can be unique for a device 101 (e.g. not recorded in other devices 101). The use of unique initiator bootstrap public keys Bi 104a recorded for responders 101x is also depicted in a key table 104k in Figure 2b. For embodiments where each device 101 records a “unique key” for the initiator bootstrap public key Bi 104a (such as during device manufacturing), then a device database 104 could record both (i) the unique initiator bootstrap public key Bi 104a for device 101 and (ii) the corresponding unique initiator bootstrap private key bi 104b (such as receiving the keys from the device manufacturer). Other possibilities exist as well, where a set of initiator bootstrap public keys 104a recorded in a device 101 for a network 107 can comprise either (i) shared keys or (ii) unique keys, without departing from the scope of the present invention.

As depicted in Figure 6b the responder 101x in device 101 can receive the message 122 from an initiator 102'. Message 122 was depicted and described in connection with Figure 3a above, and can comprise a DPP authentication request message. Responder 101x can process the message 122 and extract the hash 199e value of an initiator bootstrap public key Bi 104a. Responder 101x can (a) query 620 a table 622 with the hash 199e value H(Bi) in order to (b) select and read the value for the initiator bootstrap public key Bi 104a. The results of a query 620 can comprise a result 621 with key Bi 104a and parameters 199a. The result 621 (comprising at least one key Bi 104a as depicted in Figure 6b) can be used with a set of steps 325 depicted and described in connection with Figure 3a and Figure 5a. Steps 325 can generate a DPP authentication response 123 for initiator 102' using the selected key Bi 104a.

Although not depicted in Figure 6b, for embodiments where responder 101x records a plurality of different responder bootstrap PKI keys 101a and 101b, a query 620 can also be performed by responder 101x in order to select one responder bootstrap public key Br 101a and the corresponding responder bootstrap private key br 101b. Responder 101x could record the plurality of different responder bootstrap PKI keys in a table 623, along with a hash value 199e for each responder bootstrap public key Br 101a (not shown). Upon receipt of a message 122, responder 101x could conduct a query 620 using the received hash 199e H(Br) for a key Br 101a in a table 623 with a plurality of different responder bootstrap public keys Br 101a in order to select at least one key Br 101a and the corresponding private key br 101b. In exemplary embodiments, if the hash 199e value received in a message 122 does not match an internally calculated or recorded hash 199e value for a recorded responder bootstrap public key Br 101a, then responder 101x and device 101 remains silent on WiFi link 192, where WiFi link 192 is depicted above in Figure 3a, Figure 5a, etc.

Figure 7

Figure 7 is a graphical illustration of an exemplary system, where an access point conducts a configuration step for a device using a hosted device provisioning protocol, in accordance with exemplary embodiments. The present invention contemplates that the systems and methods for using an initiator 102' with a DPP server 103 can support a device provisioning protocol 191 in order to configure a network access point 105'. System 700 in Figure 7 can include an unconfigured network access point 105', a mobile phone 701, and a network 107. Unconfigured network access point 105' can include a responder 101x, where responder 101x can have the same or equivalent functionality as a responder 101x for a device 101 depicted and described above in Figure 1a, Figure 1b, Figure 1c, Figure 3a, Figure 5a, and Figure 5b. Responder 101x in network access point 105' can include a responder bootstrap public key Br 101a, a responder bootstrap private key br 101b, and an initiator bootstrap public key Bi 104a. PKI keys could be recorded in access point 105' before access point 105' is installed at an operating location, such as during manufacturing or distribution of unconfigured access point 105'. Responder 101x operating in unconfigured access point 105' can have responder capabilities of an enrollee.

Mobile phone 701 can comprise a smart phone or a mobile handset, such as a mobile phone manufactured by Apple® or running an Android based operating system. Mobile phone 701 could also comprise another mobile and networked computing device, such as a laptop or a tablet. Mobile phone 701 could include an initiator 102', where initiator 102' could have the same or equivalent functionality as initiator 102' depicted and described in connection with Figure 1a, Figure 1b, Figure 1c, Figure 1d, Figure 2a, Figure 3a, Figure 5a, Figure 5b, and Figure 6a. In exemplary embodiments, initiator 102' can operate as an application for mobile phone 701 downloaded from a repository of mobile phone applications, such as an "app store" or "play store", or possibly a private application. In other words, embodiments herein contemplate that a mobile phone 701 could comprise the physical entity for an initiator 102' and the software or firmware instructions for operation of initiator 102' could comprise a DPP application 102g. Other physical and/or logical arrangements for the operation of an initiator 102' are possible as well, without departing from the scope of the present invention. Mobile phone 701 can have both a WiFi radio 102w and a WAN radio 102h. Mobile phone 701 can communicate with unconfigured access point 105' via the WiFi radio 102w and a network 107 via the WAN radio 102h and an access network 112. Initiator 102' for mobile phone 701 can have capabilities of a configurator.

Network 107 can operate a DPP server 103, where a DPP server 103 in Figure 7 can have the functionality of a DPP server 103 depicted and described in connection with Figures 1a, Figure 3a, Figure 3b, Figure 5a, Figure 5b, and Figure 6a. Although not depicted in Figure 7, a network 107 can also operate other supporting servers and/or running processes depicted in Figure 1a, Figure 2a,

etc. Network 107 could be connected to mobile phone 107 via the access network 112 and an IP network 113 depicted in Figure 1a.

As depicted in Figure 7, mobile phone 107 using initiator 102' could conduct the series of messages for a device provisioning protocol 191 with responder 101x operating in unconfigured access point 105'. The series of messages for a device provisioning protocol 191 could comprise the messages 122, 123, 123a, 123b, and 124 as depicted and described in connection with Figure 3a and Figure 5a. In order for initiator 102' to send and receive the device provisioning protocol messages 191 depicted in Figure 7, initiator 102 transmit and receive a series of messages with DPP server 103 in network 107, where the series of messages could comprise a hosted device provisioning protocol 702.

For a first mode 306a, representing embodiments depicted in Figure 3a (where initiator bootstrap and ephemeral secret keys remain in DPP server 103), the series of messages for a hosted device provisioning protocol 702 could comprise the messages between initiator 102' and DPP server 103 that are depicted in Figure 3a (e.g. messages 302b, 305, 311, 317, etc.). For a second mode 306a, representing embodiments depicted in Figure 5a where the initiator bootstrap private key bi 104b can remain in DPP server 103, the series of messages for a hosted device provisioning protocol 702 could comprise the messages between initiator 102' and DPP server 103 that are depicted in Figure 5a (e.g. messages 302b, 305, 311, 503, 506, 510, etc.). For a third mode 306a, representing embodiments depicted in Figure 5b where the initiator bootstrap private key bi 104b and responder bootstrap public key Br 101a can remain in DPP server 103, the series of messages for a hosted device provisioning protocol 702 could comprise the messages between initiator 102' and DPP server 103 that are depicted in Figure 5b (e.g. messages 302b, 305, 311, 521, 522, etc.).

Upon completion of a device provisioning protocol 191 and a hosted device provisioning protocol 702, unconfigured network access point 105' can conduct a configuration step 106 to load a set of network credentials 109 and begin operating with the credentials 109, as depicted in Figure 7. The configuration step 106 can convert the unconfigured network access point 105' into an operating network access point 105, such that WiFi clients (or "sta") can connect with network access point 105 using a compatible set of network credentials 109. As depicted in Figure 7, a mobile phone 701 could also operate as an initiator 102' with a roll of a configurator for the access point 105'. Mobile phone 701 could also conduct a device provisioning protocol 191 using a hosted device provisioning protocol 702 with DPP server 103 in order to transfer a set of compatible network credentials 109 to device 101 in Figure 7, such that device 101 could connect with network access point 105'. The compatible network credentials 109 for device 101 can specify that device 101 operates in a WiFi client mode with network access point 105.

Figure 8

Figure 8 is a graphical illustration of an exemplary system, where an access point operates as an initiator for a device with a responder, in order transfer a set of network credentials to the device, in accordance with exemplary embodiments. The present invention contemplates embodiments where a network access point 105 can operate as an initiator 102' for a device 101 operating as a responder 101x, in order for the network access point 105 to transfer network credentials 109 to the device 101. System 800 in Figure 8 can include an unconfigured device 101', a network access point 105, and a network 107.

Network access point 105 in a system 800 can operate with a set of network access credentials 109 and also operate as an initiator 102'. Network access point 105 can connect with a network 107 via an access network 112. Network 107 can operate a DPP server 103, where the function and operation of a DPP server 103 is depicted and described above in connection with Figure 1a, Figure 3a, Figure 5a, etc. Although not depicted in Figure 8, network 107 could also operate a device database 104 and DPP server 103 could include a server database 103a. Network access point 105 could comprise a WiFi router and operate a WiFi radio. Network access point 105 can have the internal electrical components equivalent to an initiator 102' depicted and described above in connection with Figure 1d, including a CPU, RAM, nonvolatile memory, and a system bus. For the embodiment depicted in Figure 8, the initiator 102' in network access point 105 could comprise software, firmware, or a running process within network access point 105. Initiator 102' in network access point 105 could have the same or equivalent functionality as initiator 102' depicted and described in connection with Figure 1a, Figure 1b, Figure 1c, Figure 1d, Figure 2a, Figure 3a, Figure 5a, Figure 5b, and Figure 6a. Network access point 105 could operate with capabilities as a configurator 801, as depicted in Figure 8.

An unconfigured device 101' could be a device with a WiFi radio 101h and operate a responder 101x. The electrical components inside device 101' could be similar to the electrical components depicted for an initiator 102' in Figure 1d, such as including a CPU, RAM, nonvolatile memory, an operating system, and a system bus. Responder 101x in unconfigured device 101 could have the same or equivalent functionality as a responder 101x depicted and described above in Figure 1a, Figure 1b, Figure 1c, Figure 3a, Figure 5a, and Figure 5b. Unconfigured device 101' can have capabilities as an enrollee 802.

As depicted in Figure 8, network access point 105 initiator 102' could conduct the series of messages for a device provisioning protocol 191 with responder 101x operating in unconfigured device 101'. The series of messages for a device provisioning protocol 191 could comprise the messages 122, 123, 123a, 123b, and 124 as depicted and described in connection with Figure 3a and Figure 5a. In order for initiator 102' to send and receive the device provisioning protocol messages 191 depicted in Figure 8, initiator 102' (or AP 105 operating the initiator 102') transmits and

receives a series of messages with DPP server 103 in network 107, where the series of messages could comprise a hosted device provisioning protocol 702. The series of messages for a hosted device provisioning protocol 702 were depicted and described in connection with Figure 7 above. Note that the steps and messages for conducting a hosted device provisioning protocol 702 can depend on a mode 306a, where the mode 306a can be selected by a DPP server 103 in a step 306. The present invention also contemplates that a mode 306a for a hosted device provisioning protocol 702 can be selected by an initiator 102'. For these embodiments, where mode 306a is selected by an initiator 102', a selected mode 306a could be transmitted from initiator 102' to DPP server 103 in a message 305 depicted and described in connection with Figure 3a above.

The combined (i) device provision protocol 191 between network access point 105 and device 101 and (ii) hosted device provisioning protocol 702 between network access point 105 and DPP server 103 can securely transfer a set of compatible network access credentials 109 to device 101'. Upon completion of a device provisioning protocol 191 and a hosted device provisioning protocol 702, unconfigured device 101' can conduct a configuration step 106 to (i) load a compatible set of network credentials 109 for network access point 105 and (ii) begin operating with the credentials 109, as depicted in Figure 8. The configuration step 106 can convert the unconfigured device 101' into an operating device 101, such that device 101 can conduct a WiFi connection setup step 803 using the received set of compatible network credentials 109.

Conclusion

Various exemplary embodiments have been described above. Those skilled in the art will understand, however, that changes and modifications may be made to those examples without departing from the scope of the claims.

CLAIMS

1. A method for supporting a device provisioning protocol (DPP), the method performed by a first computing device, the method comprising:

a) establishing, by the first computing device, a secure session with a server, wherein the secure session comprises at least authentication of the first computing device;

b) receiving, by the first computing device and from a second computing device, a tag value for a responder;

c) sending, by the first computing device and to the server, at least a portion of the received tag value;

d) sending, by the first computing device and to the second computing device, an initiator nonce, wherein the initiator nonce is encrypted using at least a first shared secret derived by the first computing device;

e) receiving, by the first computing device and from the second computing device, a responder ephemeral public key for the second computing device;

f) sending, by the first computing device, the responder ephemeral public key to the server;

g) receiving, by the first computing device and from the server, a second shared secret, wherein the second shared secret is derived using at least a responder bootstrap public key and the responder ephemeral public key;

h) deriving, by the first computing device, an encryption key using at least the first shared secret and the second shared secret;

i) encrypting, by the first computing device, an authorization value using at least the derived encryption key;

j) sending, by the first computing device and to the second computing device, the encrypted authorization value; and

k) sending, by the first computing device and to the second computing device, a ciphertext of a network credential, wherein the ciphertext is encrypted using at least the derived encryption key.

2. The method of claim 1, wherein the first computing device comprises an initiator and the second computing device comprises the responder, and wherein the responder ephemeral public key comprises a responder protocol public key.

3. The method of claim 2, wherein the first computing device comprises a mobile phone operating a software application for the initiator.

4. The method of any one of claims 1 to 3, wherein the responder bootstrap public key is received from the server, and wherein the first shared secret is derived using the responder bootstrap public key and an initiator ephemeral public key.

5. The method of any one of claims 1 to 4, wherein the tag value includes the responder bootstrap public key, wherein the portion of the received tag value comprises the responder bootstrap public key, and wherein the first shared secret is derived using the responder bootstrap public key and an initiator ephemeral public key.

6. The method of any one of claims 1 to 5, wherein the initiator nonce is sent in a DPP authentication request, wherein the responder ephemeral public key is received in a DPP

authentication response, wherein the authorization value is sent in a DPP authentication confirm message, and wherein the ciphertext is sent in a DPP configuration response message.

7. The method of any one of claims 1 to 6, wherein the first shared secret is derived using (i) an initiator ephemeral private key, (ii) the responder bootstrap public key, and (iii) a first elliptic curve Diffie Hellman key exchange, and wherein the second shared secret is derived by the server using an initiator bootstrap private key and a second elliptic curve Diffie Hellman key exchange.

8. A method for supporting a device provisioning protocol (DPP), the method performed by a network, the method comprising:

a) storing, by the network, an initiator bootstrap private key and a responder bootstrap public key;

b) establishing, by the network, a secure session with a computing device, wherein the computing device authenticates with the network, and wherein the network receives a tag value for a responder via the secure session;

c) sending, by the network and to the computing device, (i) an initiator bootstrap public key for the initiator bootstrap private key, and (ii) an initiator configuration;

d) receiving, by the network and from the computing device, a responder ephemeral public key;

e) conducting, by the network, a first elliptic curve Diffie-Hellman (ECDH) key exchange using the responder bootstrap public key, the responder ephemeral public key, and the initiator bootstrap private key in order to derive a shared secret key;

f) sending, by the network and to the computing device, the derived shared secret key via the secure session; and

g) receiving, by the network and from the computing device, a plaintext configuration attribute, wherein a ciphertext configuration attribute is decrypted into the plaintext configuration attribute using at least the derived shared secret key.

9. The method of claim 8, wherein the network includes a database and a DPP server, and wherein the database stores the initiator bootstrap private key and the DPP server conducts the first ECDH key exchange.

10. The method of claim 8 or 9, wherein the computing device reads the tag value for the responder, and wherein the computing device establishes a DPP session with the responder over a WiFi connection.

11. The method of any one of claims 8 to 10, wherein the derived shared secret key comprises a value of “L” according to the device provisioning protocol.

12. The method of any one of claims 8 to 11, wherein the computing device derives an initiator ephemeral private key at least for conducting a second ECDH key exchange, wherein the second ECDH key exchange derives a value of “M” according to the device provisioning protocol, and wherein the computing device derives the value of “M” before the computing device sends a DPP authentication request message.

13. The method of claim 12, further comprising decrypting, by the computing device, the ciphertext configuration attribute using at least (i) the shared secret key received from the network, and (ii) the value of “M”.

14. A mobile device for supporting a device provisioning protocol (DPP), the mobile device comprising:

a camera for reading a tag value associated with a WiFi device;

a WAN radio for establishing a secure session with a server, for sending the tag value to the server, for receiving from the server (i) an initiator configuration (ii) a first hash value of a responder bootstrap public key and (iii) a first ciphertext;

a WiFi radio for operating with a user configuration before the mobile device reads the tag value, for operating with the initiator configuration after the mobile device reads the tag value, for transmitting a DPP authentication request message with the first ciphertext and the first hash value, and for receiving a DPP authentication response message with a second ciphertext and a responder ephemeral public key;

a nonvolatile memory for storing (i) the user configuration while the WiFi radio operates with the initiator configuration, and (ii) a DPP application for conducting the DPP over the WiFi radio;

a processor for validating the responder ephemeral public key, for encrypting the responder ephemeral public key and the second ciphertext for the secure session, and for receiving a third ciphertext from the server, wherein the third ciphertext includes data for a DPP authentication confirm message; and

a data bus for electrically connecting the camera, WAN radio, WiFi radio, nonvolatile memory, and processor, and for transferring a fourth ciphertext with a set of credentials for the WiFi device from the WAN radio to the WiFi radio.

15. The mobile device of claim 14, wherein the mobile device (i) does not store and (ii) does not receive the responder bootstrap public key for the WiFi device.

16. The mobile device of claim 14 or 15, wherein the DPP application applies the user configuration to the WAN radio after the mobile device sends the set of credentials to the WiFi device, and wherein the mobile device comprises at least one of (i) a mobile phone, (ii) a tablet computer, and (iii) a laptop computer.

17. The mobile device of any one of claims 14 to 16, wherein the WiFi device comprises a responder for the DPP, and wherein the DPP authentication request message is transmitted by the mobile device as a broadcast message over a WiFi link.

18. The mobile device of any one of claims 14 to 17, wherein the server records an initiator bootstrap private key corresponding to an initiator bootstrap public key, wherein the mobile device receives a second hash value of the initiator bootstrap public key via the secure session, and wherein the DPP authentication request message includes the second hash value.

19. The mobile device of any one of claims 14 to 18, wherein the third ciphertext is encrypted with a symmetric ciphering key, and wherein the symmetric ciphering key is derived

using an elliptic curve Diffie Hellman (ECDH) key exchange and at least (i) an initiator bootstrap private key, (ii) the responder bootstrap public key, and (iii) the responder ephemeral public key.

20. The mobile device of claim 19, wherein the WiFi radio transmits a DPP authentication confirm message that includes the third ciphertext, wherein the WiFi radio receives a DPP configuration request message after the WiFi radio transmits the DPP authentication confirm message, wherein the WiFi radio transmits a DPP configuration response message with the fourth ciphertext to the WiFi device, and wherein the fourth ciphertext is encrypted with the symmetric ciphering key.

21. A method for supporting a device provisioning protocol, the method performed by a server, the method comprising:

- a) recording, by the server, an initiator bootstrap private key;
- b) receiving, by the server and from an initiator, a responder bootstrap public key;
- c) deriving, by the server, (i) an initiator ephemeral private key, (ii) a corresponding initiator ephemeral public key, and (iii) a pseudo random number comprising an initiator nonce;
- d) conducting, by the server, (x) a first elliptic curve Diffie-Hellman (ECDH) key exchange using the responder bootstrap public key and the derived initiator ephemeral private key in order to (y) derive a first symmetric ciphering key;
- e) encrypting, by the server, the initiator nonce into a first ciphertext using the first symmetric ciphering key;
- f) sending, from the server to the initiator, the first ciphertext and the initiator ephemeral public key;

g) receiving, by the server and from the initiator, a second ciphertext and a responder ephemeral public key;

h) conducting, by the server, (x) a second ECDH key exchange using the responder ephemeral public key and the derived initiator private key in order to (y) derive a second symmetric ciphering key;

i) decrypting, by the server, the second ciphertext into a first plaintext using the second symmetric ciphering key, wherein the first plaintext from the second ciphertext includes a third ciphertext;

j) conducting, by the server, (x) a third ECDH key exchange using the received responder bootstrap public key, the received responder ephemeral public key, and the recorded initiator bootstrap private key in order to (y) derive a third symmetric ciphering key;

k) decrypting, by the server, the third ciphertext into a second plaintext using the third symmetric ciphering key, wherein the second plaintext includes a received responder authentication value;

l) determining, by the server, the received responder authentication value equals a calculated responder authentication value using at least, in part, the initiator nonce;

m) encrypting, by the server, a set of network credentials using the third symmetric ciphering key; and

n) sending, from the server to the initiator, the encrypted set of network credentials.

22. The method of claim 21, further comprising sending, from the server to the initiator, a fourth ciphertext before sending the set of network credentials, wherein the fourth ciphertext includes an initiator authentication value, and wherein the fourth ciphertext is encrypted using the third symmetric ciphering key.

23. The method of claim 21 or 22, further comprising the initiator operating as an initiator proxy, wherein the initiator proxy does not receive and does not operate with the initiator bootstrap private key.

24. The method of any one of claims 21 to 23, further comprising receiving, by a discovery server, a tag value from the initiator, wherein the discovery server uses the tag value to send the initiator a domain name for the server.

25. The method of any one of claims 21 to 24, wherein the server records a set of cryptographic parameters for the initiator bootstrap private key and the initiator ephemeral private key, wherein the server uses the set of cryptographic parameters for conducting the first, second, and third ECDH key exchanges.

26. The method of claim 25, wherein the server uses a key derivation function and the set of cryptographic parameters with the first, second, and third ECDH key exchanges in order to respectively derive the first, second, and third symmetric keys.

27. The method of any one of claims 21 to 26, further comprising establishing, by the server and the initiator, a secure session, wherein the first, second, and third ciphertexts are transmitted through the secure session.

28. A method for supporting a device provisioning protocol (DPP), the method performed by a mobile phone, the method comprising:

a) operating, by the mobile phone, a DPP application, wherein the DPP application comprises an initiator for the device provisioning protocol;

b) establishing, by the mobile phone, a secure session with a server, wherein the secure session comprises at least, in part, authentication of the DPP application;

c) receiving, by the mobile phone, a tag value for a responder;

d) sending, by the mobile phone and to the server, the received tag value;

e) receiving, by the mobile phone and from the server, an initiator configuration, wherein the mobile phone uses the received initiator configuration with a WiFi radio in the mobile phone.

f) receiving, by the mobile phone and from the server, an initiator ephemeral public key and a first ciphertext, wherein the first ciphertext includes an initiator nonce;

g) sending, by the mobile phone and to the device, the initiator ephemeral public key and the first ciphertext, wherein the mobile phone uses the received initiator configuration to send the initiator bootstrap public key and the first ciphertext to the device;

h) receiving, by the mobile phone and from the device, a responder ephemeral public key and a second ciphertext, wherein the second ciphertext includes a responder authentication value;

i) sending, by the mobile phone and to the server, the responder ephemeral public key and the second ciphertext; and

j) receiving, by the mobile phone and from the server, a third ciphertext, wherein the third ciphertext includes a set of network credentials for the device.

29. The method of claim 28, further comprising using a camera for the mobile phone in order to receive a QR code for the device, wherein the received QR code includes the tag value and

a set of cryptographic parameters, and wherein the tag value does not include the responder bootstrap public key.

30. The method of claim 28 or 29, wherein the mobile phone does not receive a responder bootstrap public key from the device, and wherein the server records the responder bootstrap public key before the mobile phone establishes the secure session with the server.

31. The method of any one of claims 28 to 30, further comprising the mobile phone conducting a radio-frequency scan in order to obtain a networks available list, wherein the mobile phone (i) sends the network available list to the server before (ii) receiving the set of network credentials in the third ciphertext.

32. The method of claim 31, further comprising the server using the networks available list in order to select a primary access network for the device, wherein the server uses the selected primary access network to select the set of network credentials from a server database.

33. The method of any one of claims 28 to 32, further comprising the mobile phone using (i) a channel list for the initiator configuration and (ii) the WiFi radio for the mobile phone to cycle through the channel list and send the initiator ephemeral public key and the first ciphertext to a responder on a plurality of channels in the channel list.

34. The method of any one of claims 28 to 33, wherein (a) sending, by the mobile phone and to the device, the initiator ephemeral public key and the first ciphertext comprises (b) a Device Provisioning Protocol (DPP) authentication request message.

35. A system for supporting a device provisioning protocol (DPP), the system comprising:

an authentication server for authenticating an initiator and for sending the authenticated initiator an initiator configuration;

a mobile phone for operating a DPP application, wherein the DPP application comprises at least, in part, the initiator, for receiving a responder bootstrap public key from a responder, wherein a secure hash value for the responder bootstrap public key is associated with a device identity, for using the initiator configuration to communicate over a WiFi link, for receiving a responder ephemeral public key using the WiFi link, for receiving an encrypted set of network credentials from a network, and for sending the encrypted set of network credentials to the responder over the WiFi link;

a device database for recording the device identity, an initiator bootstrap private key, and a set of cryptographic parameters for the bootstrap private key;

a DPP server for receiving the device identity from the initiator, for querying the device database using the device identity in order to receive the initiator bootstrap private key, for receiving the responder bootstrap public key from the mobile phone, for conducting an elliptic curve Diffie-Hellman (ECDH) key exchange using (i) the received responder bootstrap public key (ii) the received responder ephemeral public key, and (iii) the received initiator bootstrap private key, wherein the output of the ECDH key exchange comprises a symmetric ciphering key, for selecting an initiator mode for the initiator, and for sending the initiator mode to the mobile phone;

a processor in the DPP server for encrypting the set of network credentials using the symmetric ciphering key and the set of cryptographic parameters; and

a LAN interface in the DPP server for sending the encrypted set of network credentials to the mobile phone over a secure connection established between the DPP server and the mobile phone.

36. The system of claim 35, wherein the mobile phone operates as an initiator proxy, and wherein the mobile phone does not receive and does not operate with the initiator bootstrap private key.

37. The system of claim 35 or 36, further comprising a responder for recording an initiator bootstrap public key corresponding to the initiator bootstrap private key, and for using the recorded initiator bootstrap public key and to derive the symmetric ciphering key,

38. The system of claim 37, further comprising a network access point for using the set of network access credentials, and for providing connectivity with an IP network to the responder using the set of network access credentials

39. The system of any one of claims 35 to 38, wherein the mobile phone receives the responder ephemeral public key in a DPP authentication response message, wherein the DPP authentication response message includes a ciphertext of a responder authentication value, and wherein the ciphertext is encrypted with the symmetric ciphering key.

40. The system of any one of claims 35 to 39, wherein the DPP server uses the initiator bootstrap private key with a plurality of different responder bootstrap public keys, wherein the

plurality of different responder bootstrap public keys are associated with a plurality of different responders.

41. A method for supporting a device provisioning protocol (DPP), the method performed by a computing device, the method comprising:

a) recording, by the computing device, (i) a plurality of responder bootstrap private keys, and (ii) a plurality of initiator bootstrap public keys, wherein at least the plurality of initiator bootstrap public keys is recorded during one of a manufacturing and a distribution of the computing device;

b) operating, by the computing device, a WiFi radio and a responder, wherein the WiFi radio communicates with an initiator proxy;

c) receiving a DPP authentication request message with (i) a first secure hash value of a responder bootstrap public key and (ii) a second secure hash value of an initiator bootstrap public key;

d) selecting (i) a responder bootstrap private key from the plurality of responder bootstrap private keys using the first secure hash value and (ii) the initiator bootstrap public key from the plurality of initiator bootstrap public keys using the second secure hash value, wherein a DPP server records a corresponding initiator bootstrap private key for the initiator bootstrap public key;

e) generating a DPP authentication response using at least the responder bootstrap private key and the initiator bootstrap public key; and

f) receiving a DPP configuration response with network credentials for an access point.

42. The method of claim 41, wherein the plurality of initiator bootstrap public keys is recorded before the computing device is located with the access point.

43. The method of claim 41 or 42, wherein the computing device is associated with a tag, and wherein the initiator proxy reads the tag in order to receive the DPP authentication request message from the DPP server.

44. The method of any one of claims 41 to 43, further comprising:
recording, by the computing device a corresponding plurality of responder bootstrap public keys;
selecting a responder bootstrap public key from the plurality of responder bootstrap public keys using the first secure hash value;
generating the DPP authentication response using at least the responder bootstrap public key; and
an authentication server for authenticating an initiator and for sending the authenticated initiator an initiator configuration.

45. A method for supporting a device provisioning protocol (DPP), the method performed by a network, the method comprising:

- a) recording a responder bootstrap public key, a set of cryptographic parameters, and an identity token for a device in a device database;
- b) authenticating a mobile handset, wherein the network receives the identity token for a device from the authenticated mobile handset;

- c) selecting the responder bootstrap public key and the set of cryptographic parameters using the identity token;
- d) deriving, by the network and using the set of cryptographic parameters, (i) an initiator ephemeral private key, (ii) a corresponding initiator ephemeral public key, and (iii) a pseudo random number comprising an initiator nonce;
- e) conducting, by the network, (x) a first elliptic curve Diffie-Hellman (ECDH) key exchange using the responder bootstrap public key and the derived initiator ephemeral private key in order to (y) derive a first symmetric ciphering key k1;
- f) encrypting, by the network, at least the initiator nonce into a first ciphertext using the first symmetric ciphering key k1;
- g) sending, from the network and to the mobile handset, the first ciphertext and the initiator ephemeral public key;
- h) receiving, by the network and from the mobile handset, a second ciphertext and a responder ephemeral public key;
- i) conducting, by the network, (x) a second ECDH key exchange using the responder ephemeral public key and the derived initiator private key in order to (y) derive a second symmetric ciphering key k2;
- j) decrypting, by the network, the second ciphertext into a first plaintext using the second symmetric ciphering key k2, wherein the first plaintext from the second ciphertext includes a third ciphertext;
- k) conducting, by the network, (x) a third ECDH key exchange using the recorded responder bootstrap public key and the received responder ephemeral public key in order to (y) derive a third symmetric ciphering key ke;

l) decrypting, by the network, the third ciphertext into a second plaintext using the third symmetric ciphering key ke , wherein the second plaintext includes a received responder authentication value;

m) determining, by the network, the received responder authentication value equals a calculated responder authentication value using at least, in part, the initiator nonce;

n) encrypting, by the network, a set of network credentials for the device using the third symmetric ciphering key ke ; and

o) sending, from the network and to the mobile handset, the encrypted set of network credentials.

46. The method of claim 45, wherein the network receives the responder bootstrap public key from at least one of (i) a device manufacturer and (ii) a device owner.

47. The method of claim 45 or 46, wherein the identity token comprises a secure hash value of the responder bootstrap public key.

48. The method of claim 45 or 46, wherein the identity token comprises a device identity for the device.

49. The method of any one of claims 45 to 48, wherein the network and the mobile handset establish a secure session after the mobile handset authenticates with the network, and wherein the first, second, and third ciphertexts are transmitted through the secure session.

50. The method of any one of claims 45 to 49, wherein the network comprises at least an authentication server, the device database, and a DPP server.

51. The method of any one of claims 45 to 50, wherein the device operates a responder for the device provisioning protocol, and wherein the device (i) mutually derives the third symmetric ciphering key ke and (ii) decrypts the encrypted set of network credentials using the third symmetric ciphering key ke .

52. The method of any one of claims 45 to 51, wherein the device does not record an initiator bootstrap public key for the network.

53. The method of any one of claims 45 to 52, wherein the network receives the set of network credentials for the device from the authenticated mobile handset.

54. The method of any one of claims 45 to 52, wherein the network receives the set of network credentials from a device owner.

55. The method of any one of claims 45 to 54, wherein the initiator ephemeral public key comprises an initiator protocol public key, and wherein the responder ephemeral public key comprises a responder protocol public key.

Figure 1a

100

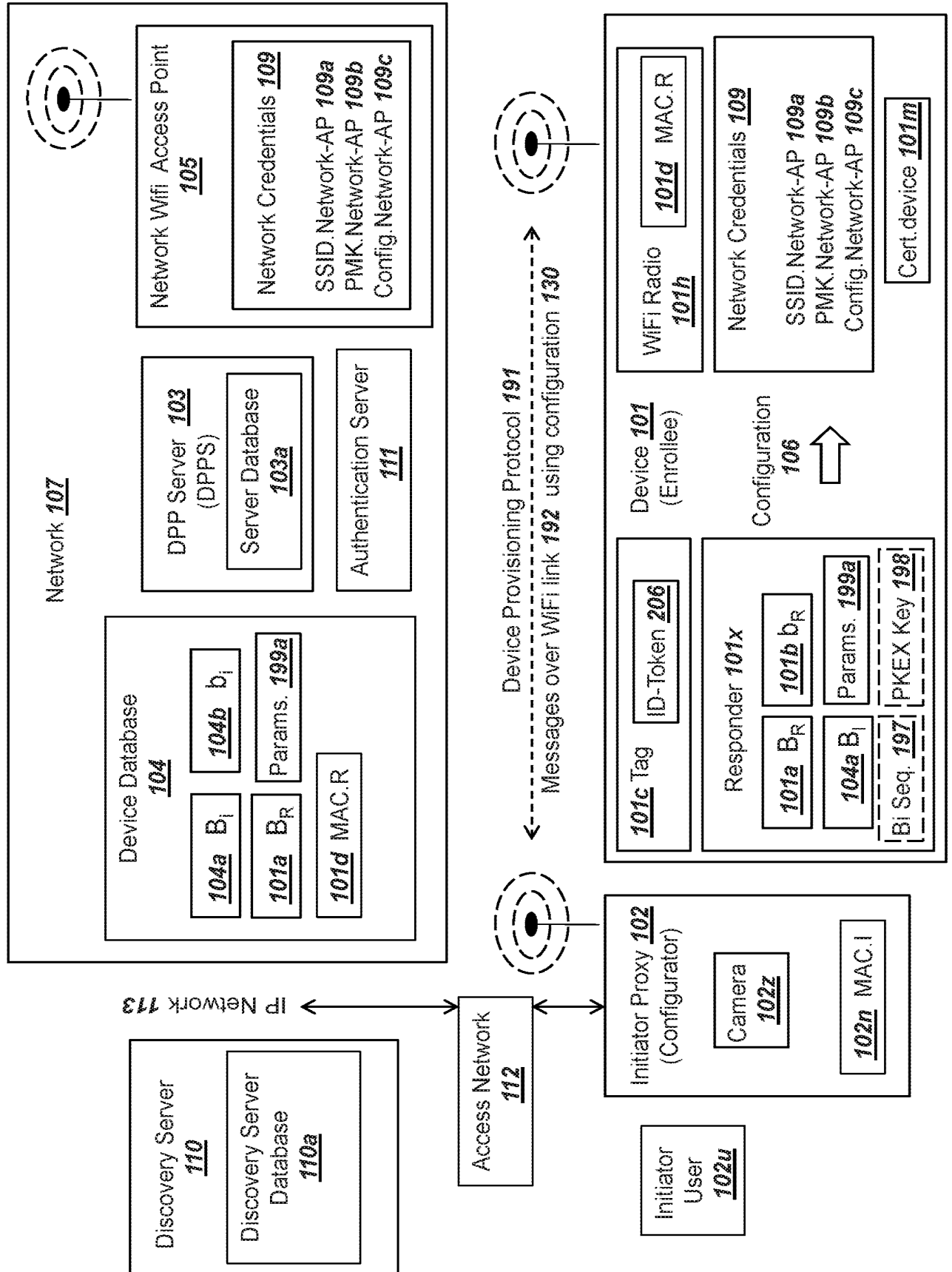


Figure 1b

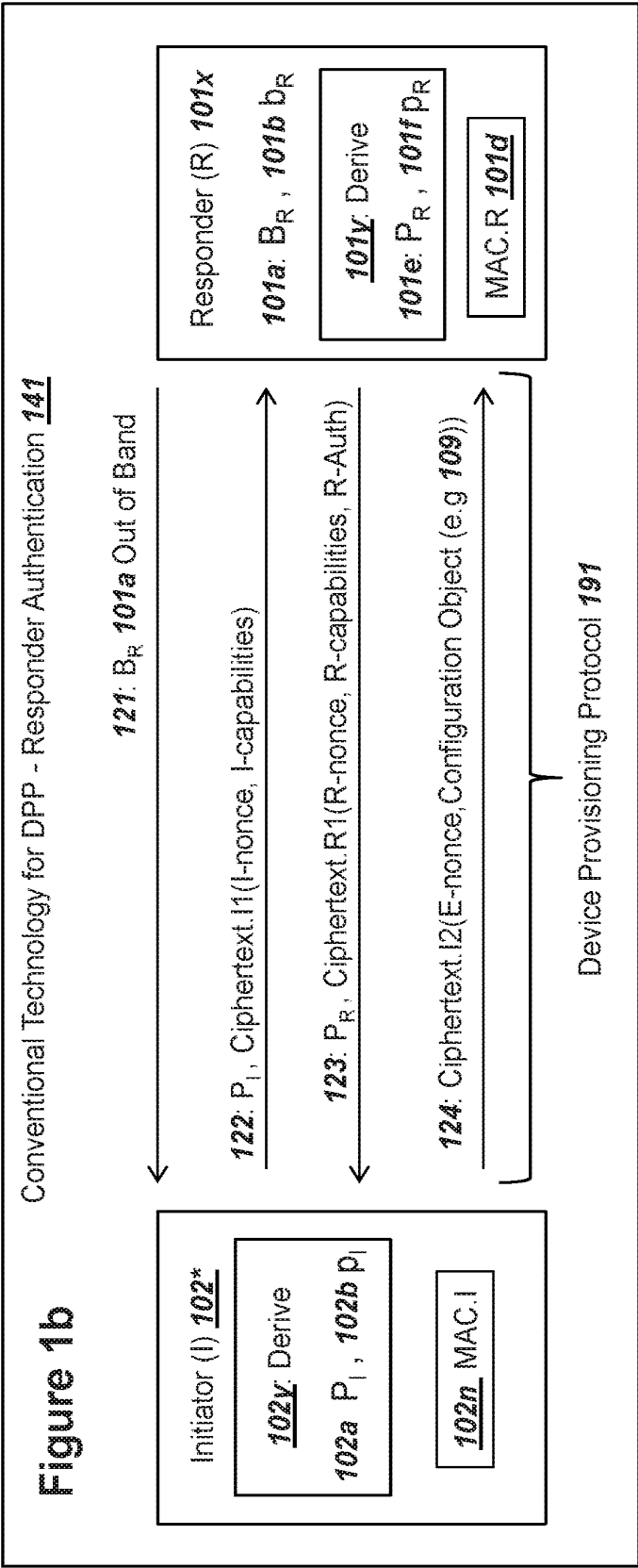


Figure 1c

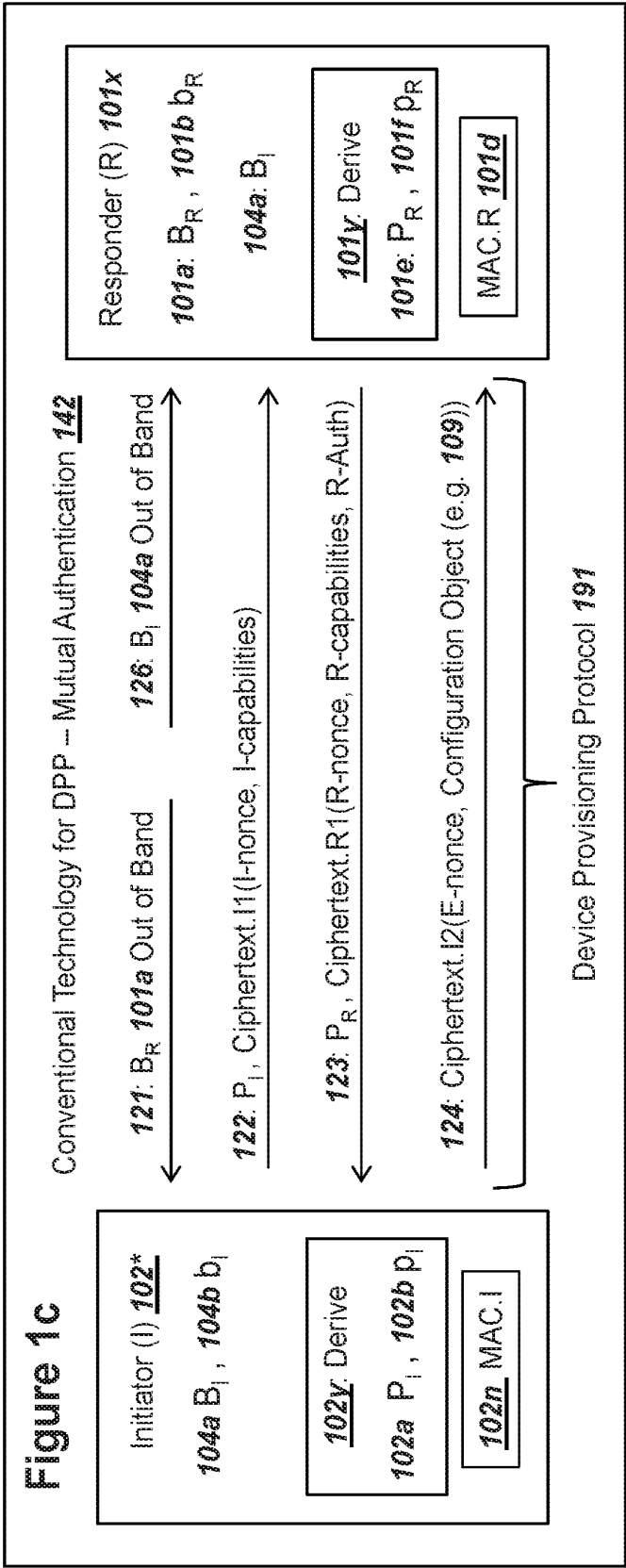
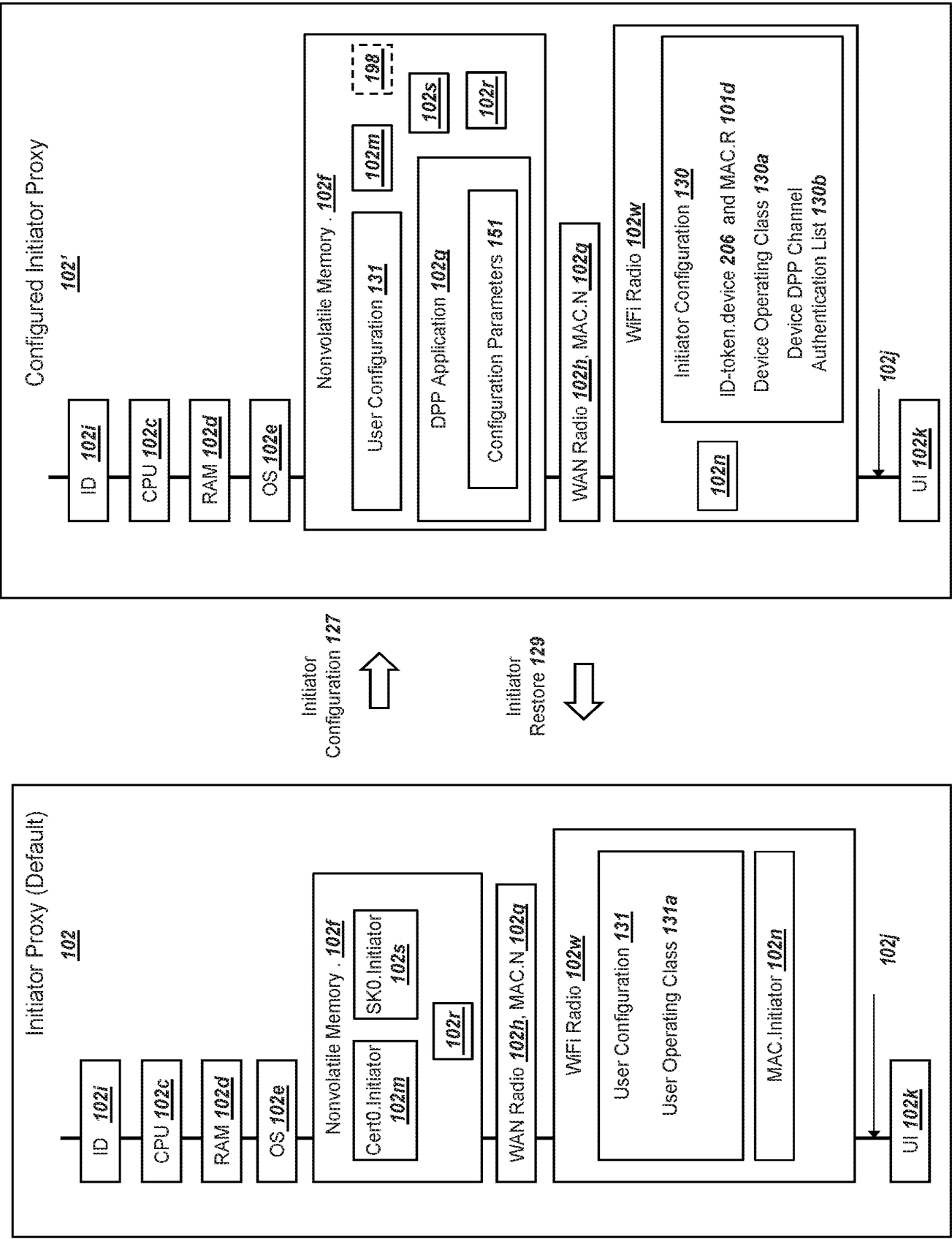


Figure 1d



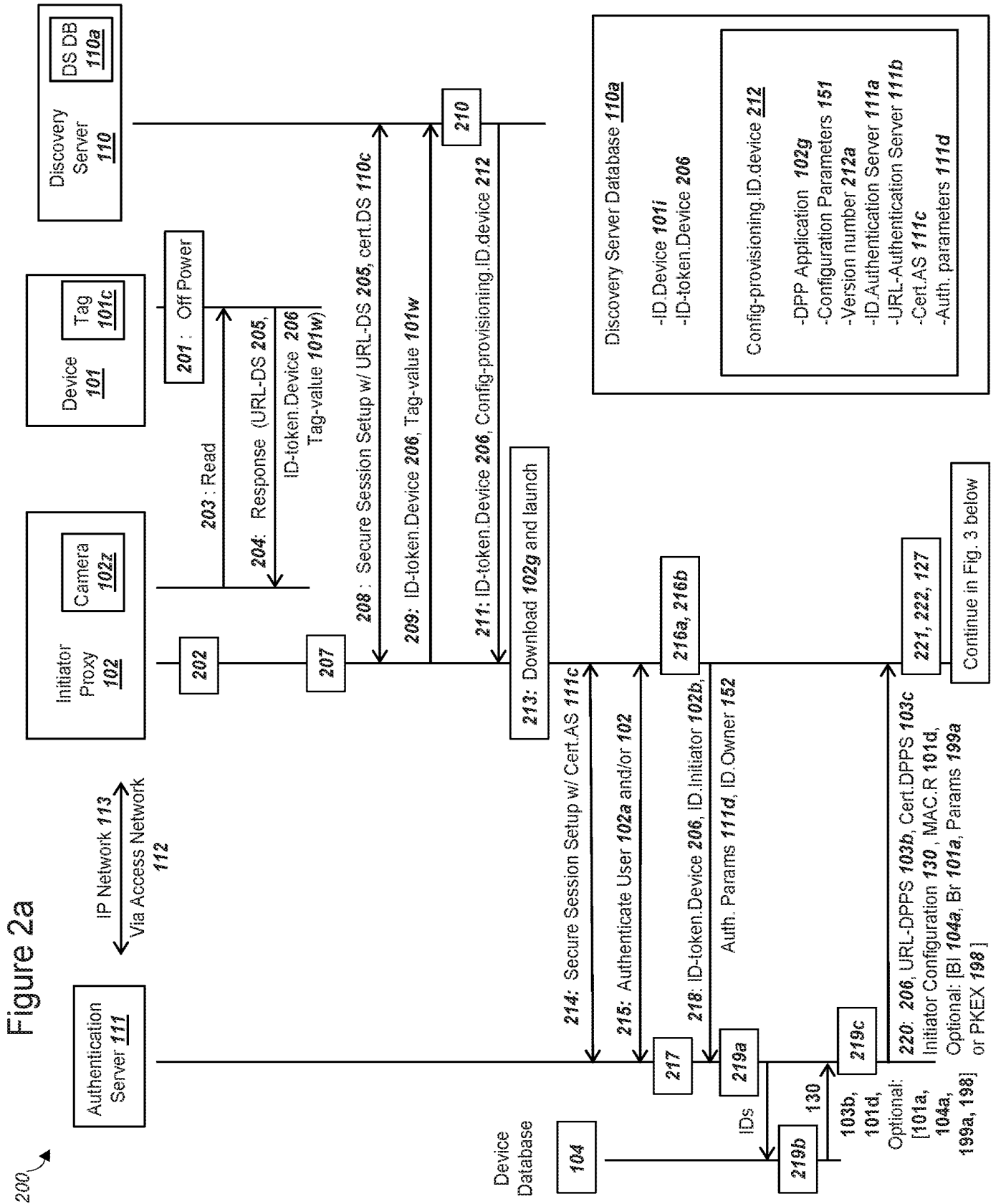


Figure 2b

Device Database 104

130

ID-token. Device	MAC. Responder	Global Operating Class	Channel List	Cryptog. Params
206 101i	101d	130a	130b	199
IRLRNZXW 20001	38607720A	81	12, 13	A
QFHXKNC 20002	386077209	83	2, 7, 9	A
KOWXGVS 20003	386077400	84	5, 6	A, B
VPMJSLMV 20004	3860773cc	117	40	A, C
WKYPUABJ 20005	386077a09	125	149, 153	A, D

Key Table 104k

PKEX Key	Responder Bootstrap Public Key 101a	Initiator Bootstrap Public Key 104a	Initiator Bootstrap Private Key 104b
X1	Br-1	Bi-1 (Shared Keys)	bi-1
X2	Br-2	Bi-1	bi-1
NA	Br-3(A), Br-3(B)	Bi-3(A), Bi-3(B)	bi-3(B)
X4	Br-4(A), Br-4(C)	Bi-4(A), Bi-4(C)	bi-4(C)
X5	Br-5(A), Br-5(D)	Bi-5(A), Bi-5(D)	bi-5(D)

(Unique Keys)

Cryptographic Parameters 199

Parameters Table 104p

Set 199a	Key Length 199b	Curve 199c	AES-SIV		Nonce Length 199g
			Key Length 199f	Hash Length 199e	
A	256	p256	SHA-256	256	128
B	384	p384	SHA-384	384	192
C	512	secp521r1	SHA-512	512	256
D	384	Curve25519	SHA-3	384	192

Device Database 104

Device Table 104d

ID-token. Device	Initiator Bootstrap Keys Seq. # 197	PKEX Key 198	Responder Bootstrap Public Key 101a	Initiator Bootstrap Public Key 104a	Initiator Bootstrap Private Key 104b
IRLRNZXW 206	1	X1a	Br-1	Bi-1a	bi-1a
IRLRNZXW	2	X1b	Br-1	Bi-1b	bi-1b
IRLRNZXW	3	X1c	Br-1	Bi-1c	bi-1c
IRLRNZXW	4	X1c	Br-1	Bi-1d	bi-1d

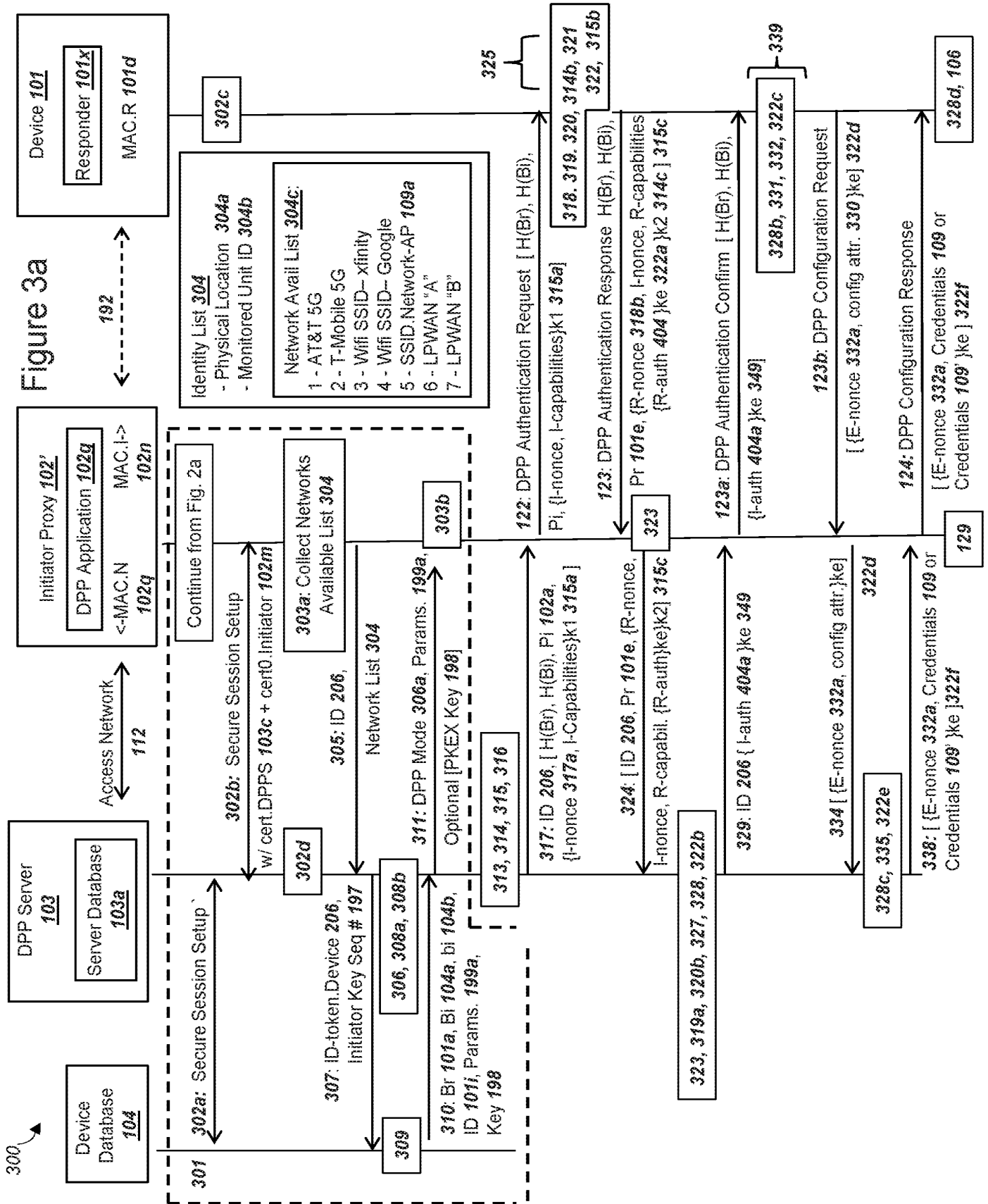


Figure 3b

Server Database 103a												
ID-token. Device 206	MAC. Responder 101d	MAC. Initiator 102q	Initiator Mode 306a	Initiator Cryptog. Params 199a	Initiator		Resp.		Initiator		PKEX	
					Ephem. Public Key 102a	Ephem. Private Key 102b	Ephem. Public Key 101e	Ephem. Private Key 101f	Bootstrap Keys Seq. # 197	Bootstrap keys 399	Shared Secret k1 314a	Shared Secret k2 314c
IRLRNZXW	38607720A	6cf049080	Fig. 3	A	PI-1	pi-1	Pr-1	Pr-1	4	Br-1, Bi-1, bi-1	X1	ke-1
QFIXHKNC	386077209	1c6f654d2	Fig. 5a	A	-	-	Pr-2	Pr-2	1	Br-2, Bi-1, bi-1	NA	-
KOWXGVTS	386077400	7071bce29	Fig. 5b	B	-	pi-3	Pr-3	Pr-3	2	{Br-3, Bi-3, bi-3} B	NA	-
VPMJSLMV	3860773cc	7071bce29	Fig. 3	A	PI-4	pi-4	Pr-4	Pr-4	1	{Br-4, Bi-4, bi-4} A	X4	ke-4
WKYPUBUJ	386077a09	7071bce29	102*	D	-	-	-	-	1	{Br-5, Bi-5, bi-5} D	X5	-
● ● ●												

Device Credentials Table 340				
ID- token.Device 206	Selected Network (Primary) 308aa	Selected Network (Backup) 308ab	Primary Network Credentials 109	Backup Network Credentials 109
IRLRNZXW	109a - 1	304a - 3	109 - 1	From xFinity
QFIXHKNC	109a - 1	304a - 6	109 - 1	From LPWAN A
KOWXGVTS	304c - 2	109a - 1	From T-Mobile	109 - 1
VPMJSLMV	304c - 4	109a - 2	From Google	109 - 2
WKYPUBUJ	109a - 3	304a - 4	109 - 3	From Google
● ● ●				

Figure 4a

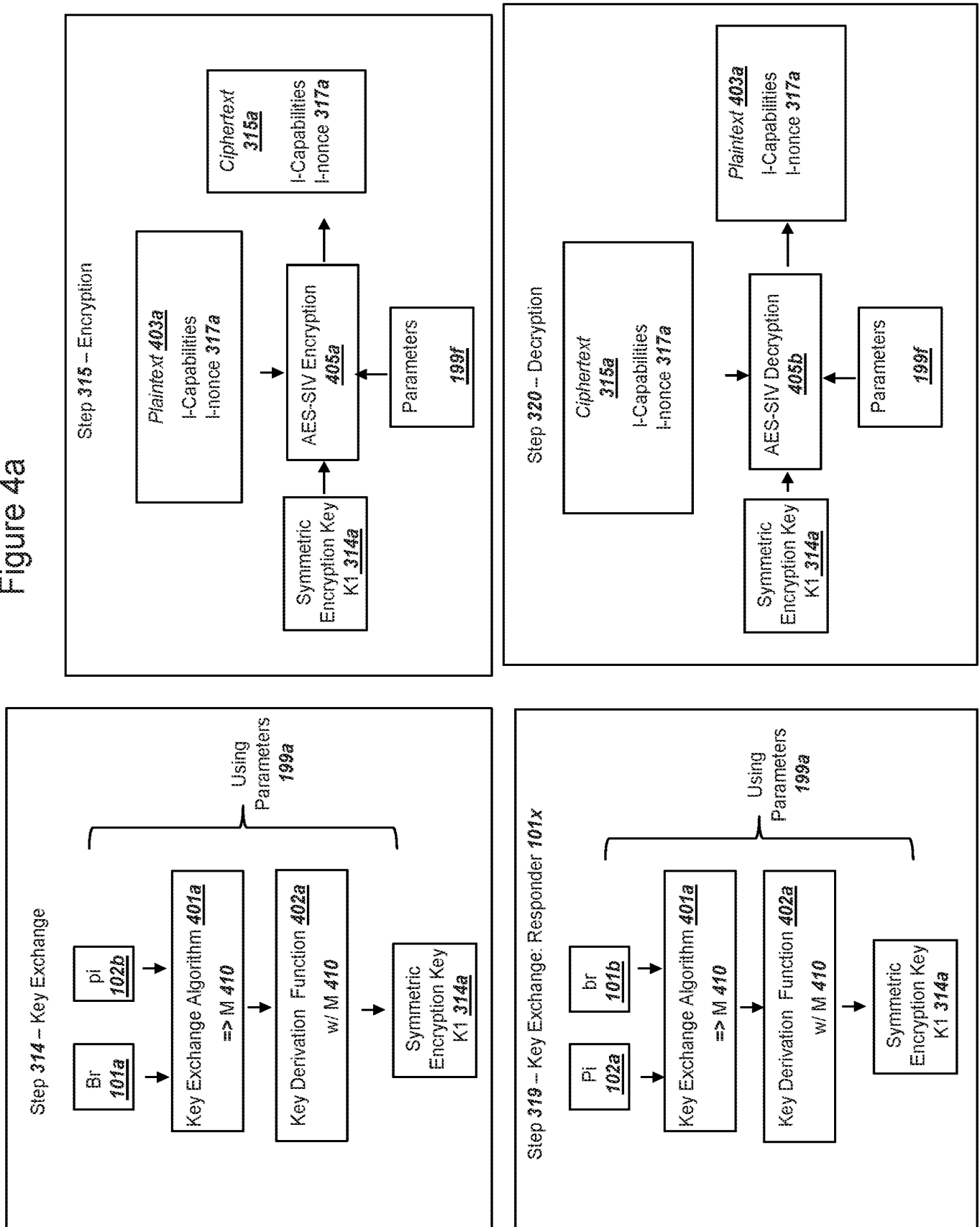


Figure 4b

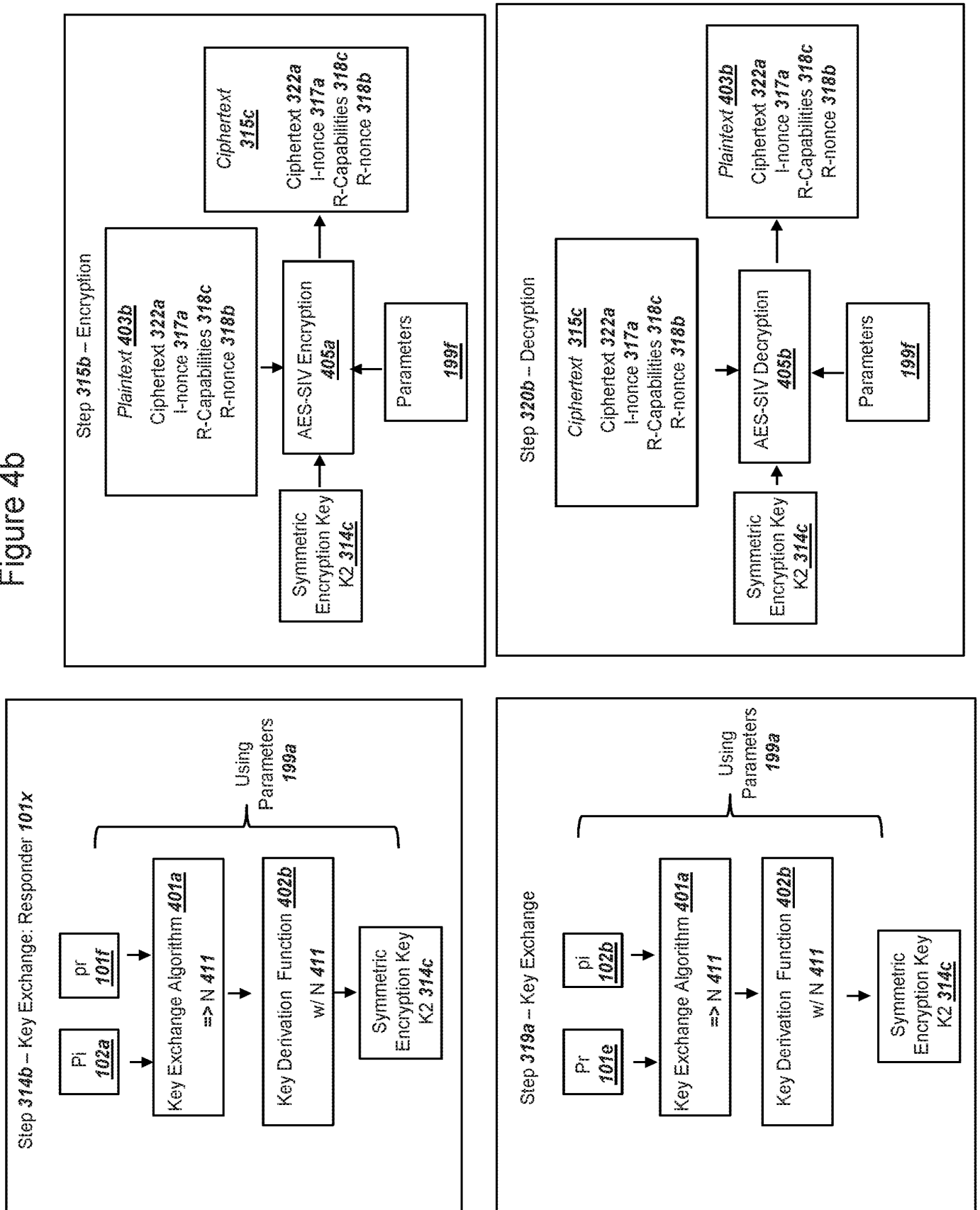


Figure 4c

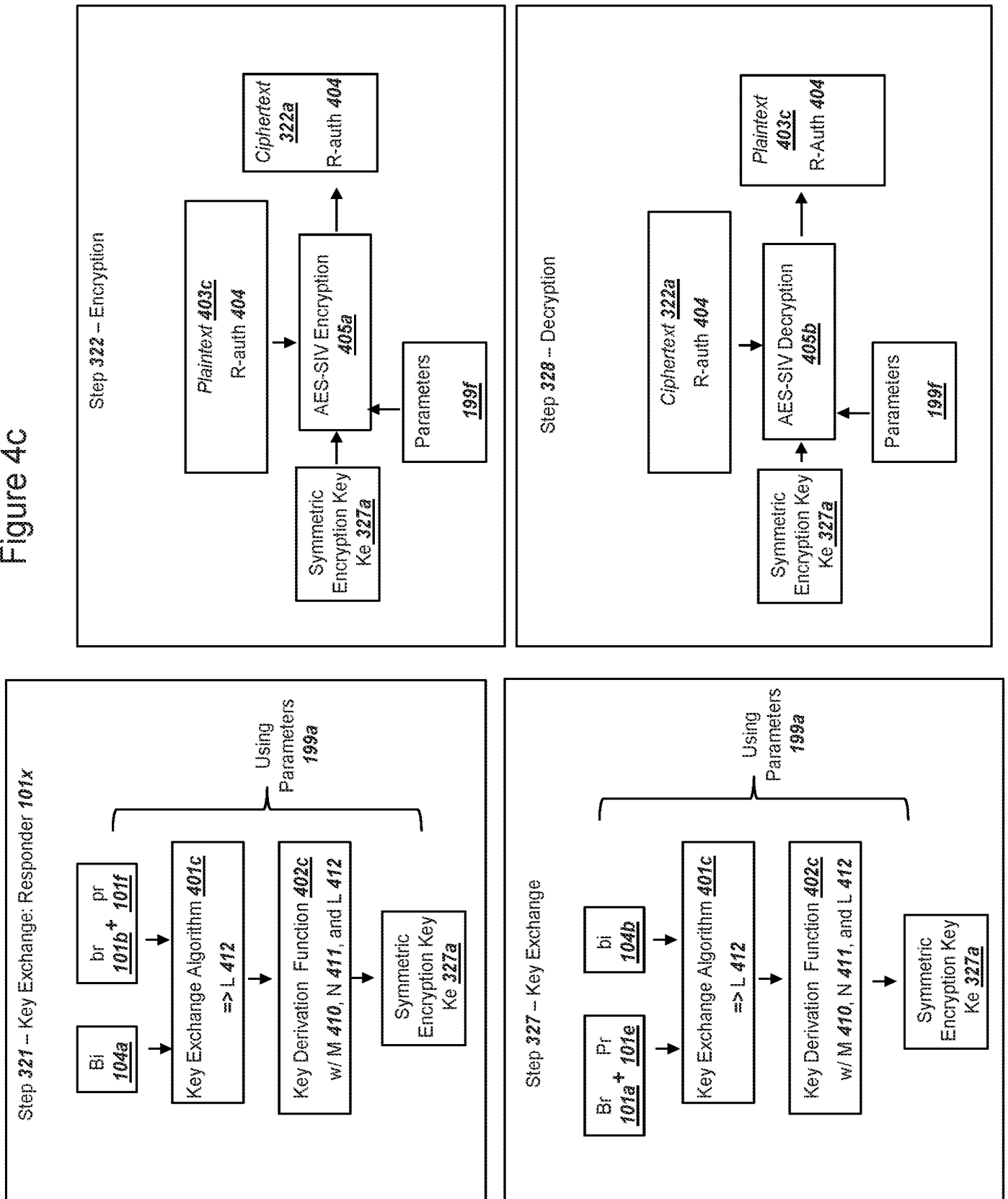
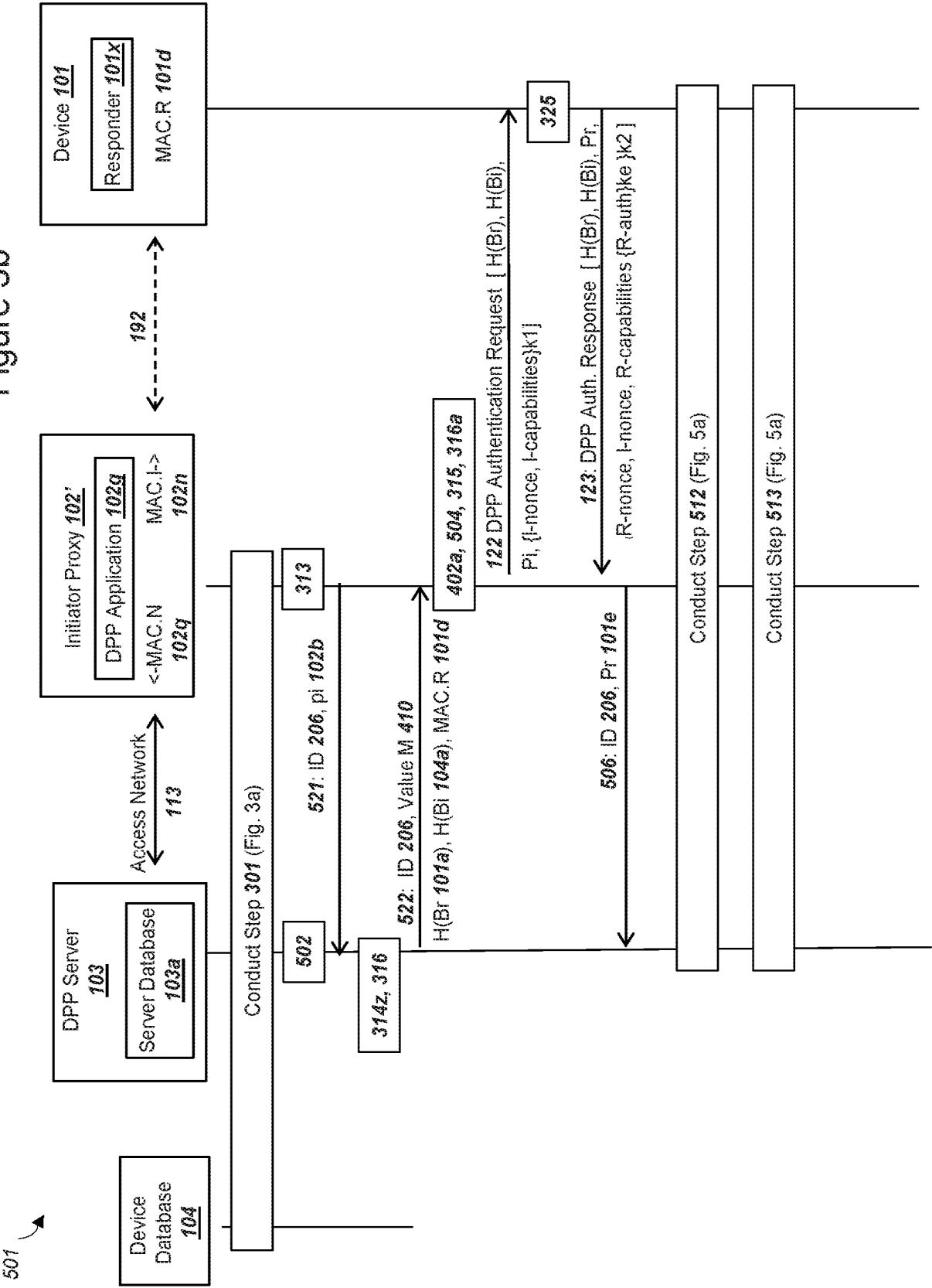
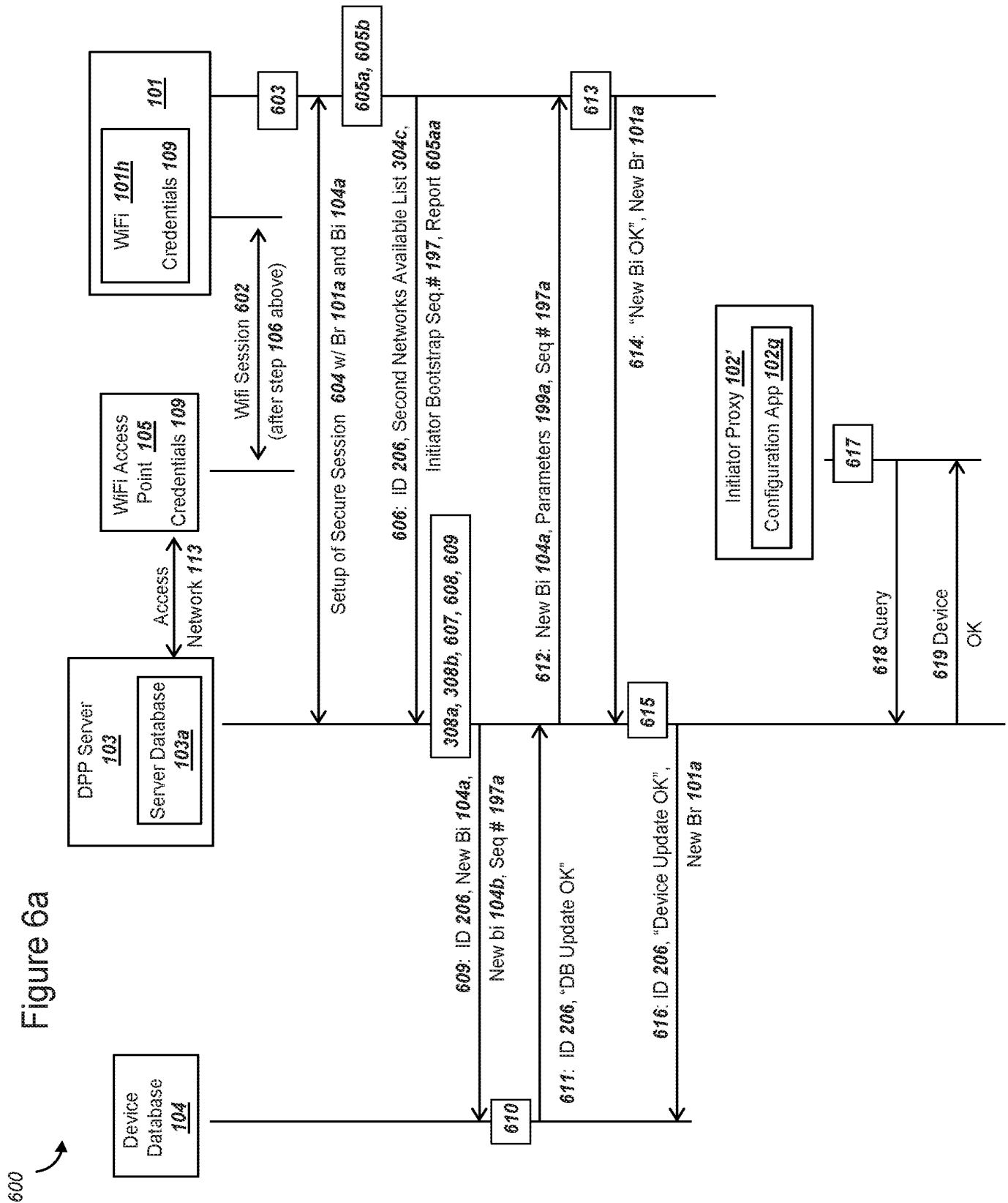


Figure 5b





601
Figure 6b

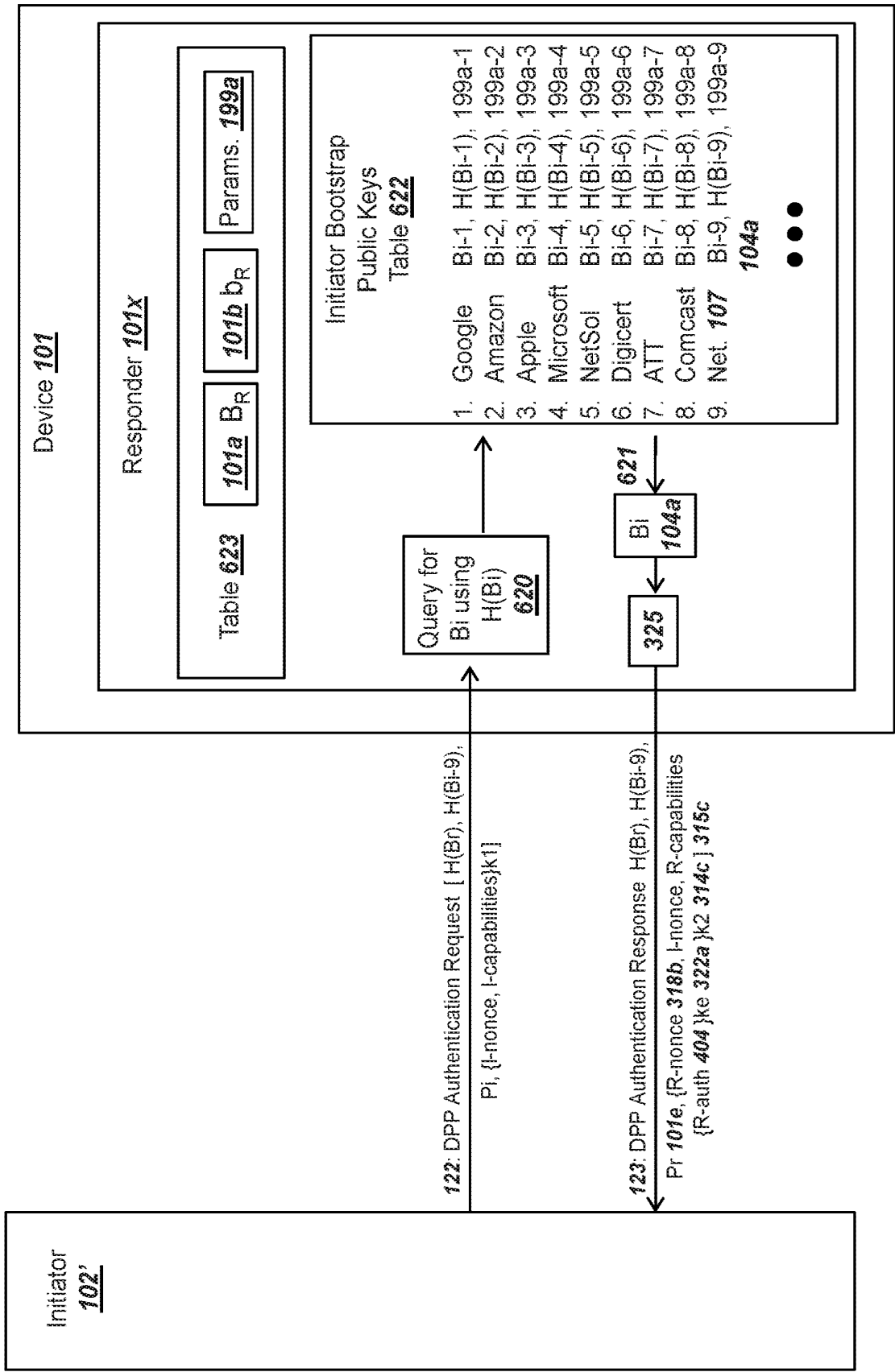


Figure 7

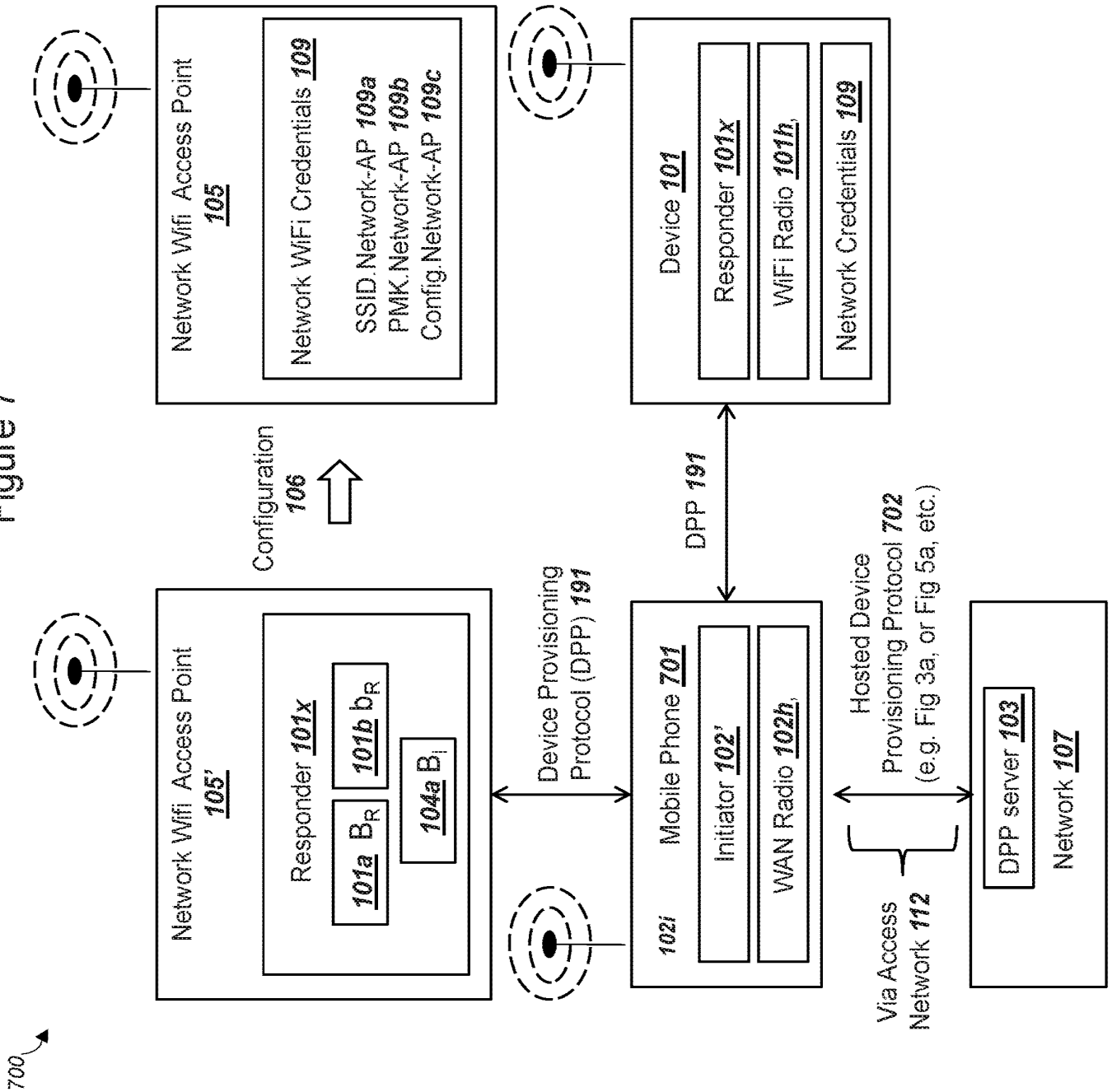
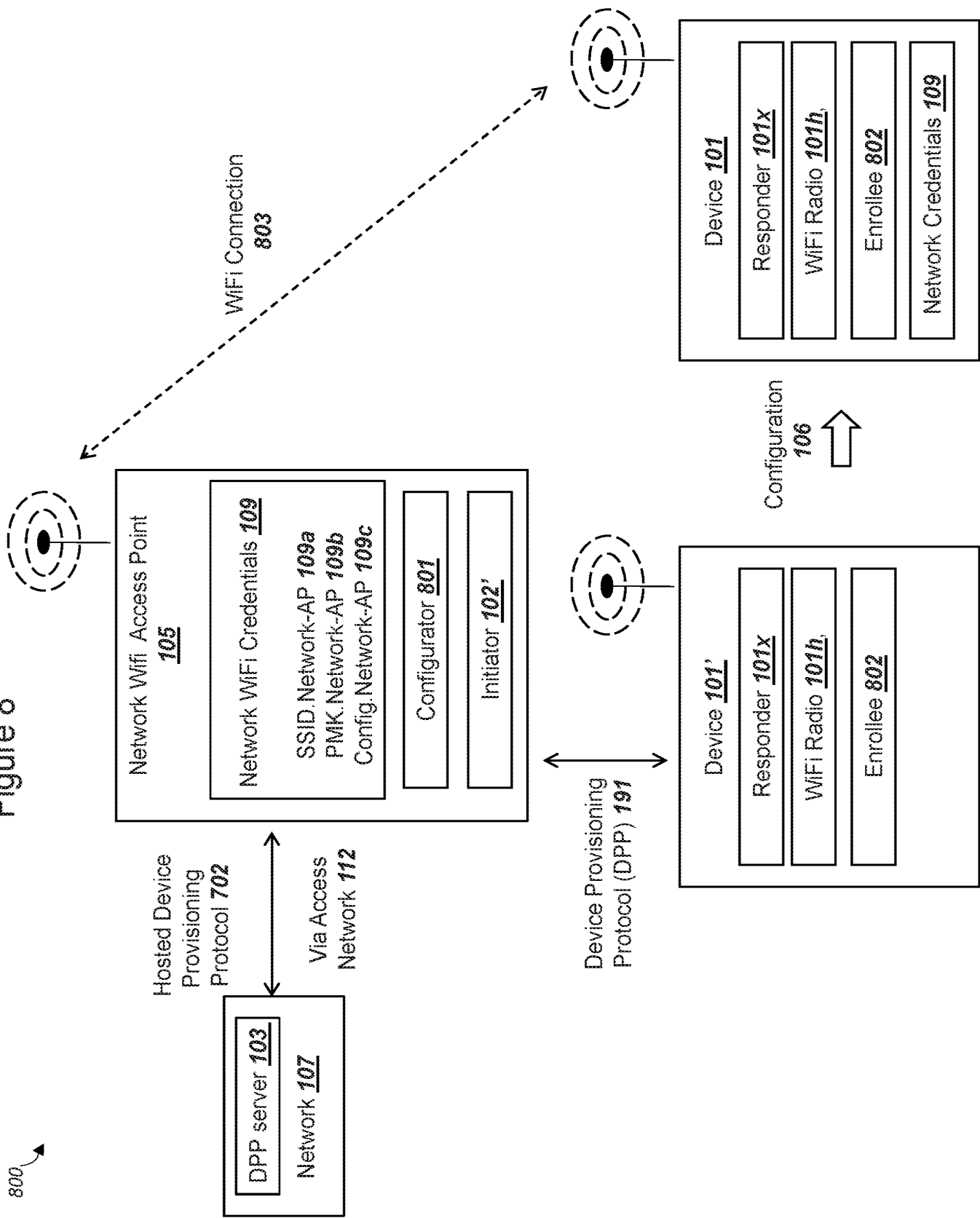


Figure 8



INTERNATIONAL SEARCH REPORT

International application No
PCT/US2019/027968

A. CLASSIFICATION OF SUBJECT MATTER

INV. H04L9/08 H04L9/32 H04L29/06 H04W12/06 H04W12/08
H04L29/08 H04W12/00 G06F21/62 H04L9/14 H04W12/04
H04W84/12 H04W88/02

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04W H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	<p>ANONYMOUS: "Wi-Fi Alliance Device Provisioning Protocol (DPP) - Draft Released for Public Review and Comments v0.0.23", WIFI ALLIANCE, , vol. v0.0.23 26 July 2016 (2016-07-26), pages 1-89, XP009501333, Retrieved from the Internet: URL:https://www.wi-fi.org/downloads-registered-guest/Wi-Fi_DPP_Tech_Spec_v0_0_23.pdf pages 24 and 45-50</p> <p style="text-align: center;">----- -/--</p>	1-55



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

8 July 2019

Date of mailing of the international search report

16/07/2019

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Jardak, Christine

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2019/027968

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Wikipedia: "Hash table - Wikipedia", 31 January 2018 (2018-01-31), pages 1-17, XP055602768, Retrieved from the Internet: URL:https://en.wikipedia.org/w/index.php?title=Hash+table&oldid=823310127 [retrieved on 2019-07-05] page 1 -----	41-44
A	US 2018/109418 A1 (CAMMAROTA ROSARIO [US] ET AL) 19 April 2018 (2018-04-19) figures 4,7 -----	1-55
A	US 2018/109381 A1 (CAMMAROTA ROSARIO [US] ET AL) 19 April 2018 (2018-04-19) paragraph [0064] - paragraph [0071]; figures 5,6 -----	1-55

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2019/027968

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2018109418 A1	19-04-2018	US 2018109418 A1	19-04-2018
		WO 2018075198 A1	26-04-2018
US 2018109381 A1	19-04-2018	CA 3036803 A1	26-04-2018
		CN 109863770 A	07-06-2019
		US 2018109381 A1	19-04-2018
		WO 2018075135 A1	26-04-2018

(51) International Patent Classification:
G06F 21/64 (2013.01) *H04L 9/14* (2006.01)(21) International Application Number:
PCT/US2019/066257(22) International Filing Date:
13 December 2019 (13.12.2019)

(25) Filing Language: English

(26) Publication Language: English

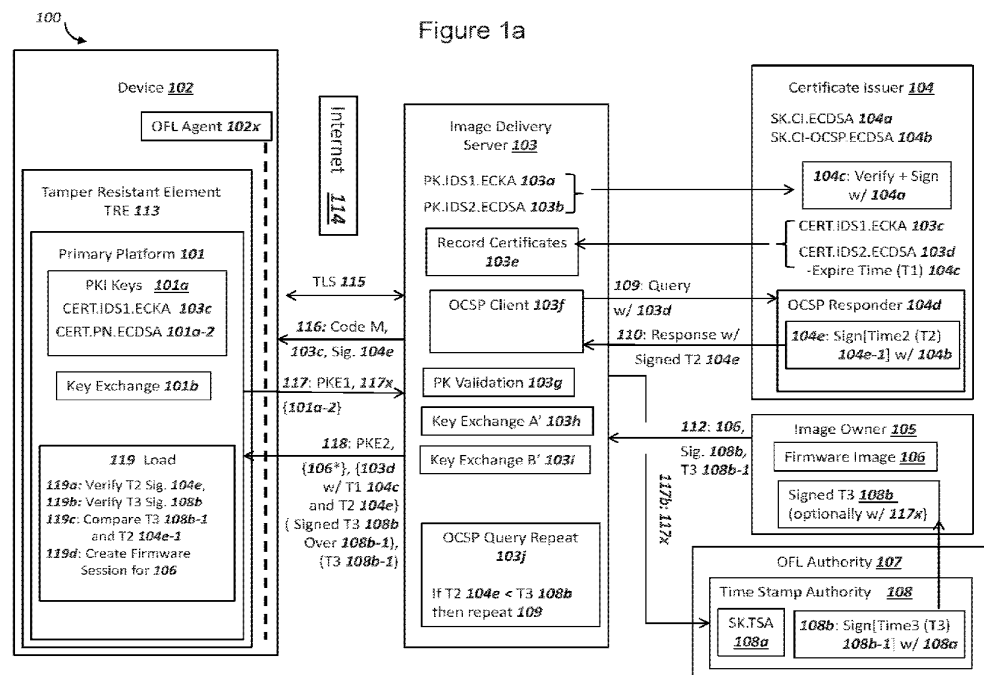
(30) Priority Data:
62/779,791 14 December 2018 (14.12.2018) US(71) Applicant: IOT AND M2M TECHNOLOGIES, LLC
[US/US]; 807 Davis Street, Unit 2207, Evanston, IL
60201-7104 (US).

(72) Inventor: NIX, John, A.; c/o IoT and M2M Technologies, LLC, 807 Davis Street, Unit 2207, Evanston, IL 60201-7104 (US).

(74) Agent: THOMSON, Kirsten, L.; McDonnell Boehnen Hulbert & Berghoff LLP, 300 South Wacker Drive, Chicago, IL 60606 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(54) Title: SECURE IDS CERTIFICATE VERIFICATION FOR A PRIMARY PLATFORM



(57) **Abstract:** A tamper resistant element (TRE) in a device can operate a primary platform and support a "Smart Secure Platform". The TRE may not keep time when electrical power is removed from the TRE. The device can receive (i) a certificate for an image delivery server (IDS) with a first timestamp and (ii) a signed second timestamp from a certificate authority, comprising a signature according to the Online Certificate Status Protocol (OCSP) with stapling. The device can forward the certificate and second timestamp to the TRE. The device can receive a ciphertext and an encrypted image from the IDS, where the ciphertext includes a third timestamp from a Time Stamp Authority (TSA), and forward the data to the TRE. The TRE can conduct a key exchange to decrypt the ciphertext. The TRE can compare the second and third timestamps to verify the certificate has not been revoked.

[Continued on next page]

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

- *with international search report (Art. 21(3))*
- *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

SECURE IDS CERTIFICATE VERIFICATION FOR A PRIMARY PLATFORM**CROSS-REFERENCE TO RELATED APPLICATION**

This international PCT application claims the benefit of the filing date of U.S. Provisional Patent Application Serial No. 62/779,791, filed December 14, 2018, which is hereby incorporated by
5 reference in its entirety.

BACKGROUND**Technical Field**

The present systems and methods relate to securely transferring firmware to a primary
10 platform from a server, and more particularly for a tamper resistant element (TRE) to determine if a certificate or public key associated with an image delivery server has been revoked.

Description of Related Art

The commercial development and deployment of secure processing environments within
15 microprocessors and tamper resistant elements (TRE) can increase security of computing devices such as mobile phones and networked sensors for the “Internet of Things”. A secure processing environment or TRE can enable a device to securely record cryptographic keys and conduct cryptographic operations such as key exchanges, key derivation, and also digital signatures in a computing environment that is isolated from other electrical components within the device that may
20 not be secure. Examples available today include an embedded subscriber identity module (embedded SIM), which is also known as an embedded universal integrated circuit card (eUICC), a traditional SIM card, a secure enclave within a “system on a chip”, a trusted execution environment (TEE), and other examples are available as well. A common computing architecture includes a processor with multiple cores, where a secure processing core is isolated from the other cores, and
25 the secure processing core can read cryptographic keys and conduct cryptographic operations. Other possibilities exist as well, including “switching” a generic processor between an insecure mode to a secure mode.

As the number of transistors and memory cells available for a given surface area of silicon continues to grow, the computational power for secure processing environments continues to
30 increase. Some secure processing environments can operate as a host computing environment and provide the functionality of a virtual machine for different firmware images, such that each firmware

image can operate as separate computing environments within the secure processing environment. As one example, secure processing environments can now enable a TRE to operate as a primary platform for hosting potentially a variety of different firmware images, where each of the firmware images can support a different application. As one example, a secure processing environment could
5 operate a Java™ virtual machine, and different firmware images could comprise different Java applets or Java-based applications. Each Java applet could comprise different images for the Java virtual machine. A firmware can also comprise the base operating system for a primary platform, such as a primary firmware of machine executable program instructions for the primary platform to operate as a host for secondary firmware to operate as virtual machines. Other possibilities exist as
10 well for a secure processing environment to operate as a primary platform for hosting images, and the images may also be referred to as a firmware image.

A primary platform both (i) operating within a secure processing environment and (ii) using firmware images can support a variety of useful applications for a computing device. The operation of an example primary platform can perform functions outlined in “Open Firmware Loader for
15 Tamper Resistant Element”, version 1.3 from Global Platform dated June 2017, which is hereby incorporated by reference in its entirety (“GP OFL document”). An open firmware loader can be referred to as “OFL”. The operation of an example primary platform can also perform functions outlined in the GSM Association (GSMA) technical document “iUICC POC Group Primary Platform requirements”, Release 1.0 dated May 17, 2017, which is hereby incorporated by reference
20 in its entirety (“GSMA PP Requirements”). Example applications supported by a primary platform using firmware images are identified in Section 3 “Use Cases” of the GSMA PP Requirement. Example applications include firmware for an embedded SIM, a traditional universal integrated circuit card (UICC), mobile payments, secure bootstrapping, digital rights management, user identification such as a drivers license, secure access to home automation, a virtual automobile key,
25 and other applications are identified as well.

Further the European Telecommunications Standards Institute (ETSI) has begun developing standards for a “Secure Primary Platform” (SSP) as part of the development of 5G standards, and an SSP could operate as a primary platform as well. As of December, 2018, the draft standards for an SSP are not available for public review, but will likely support applications similar to those
30 contemplated in the GSMA PP requirements document and GP OFL document. As the standards for operating a primary platform or a secure primary platform (SSP) continue to evolve, new features or the use of new cryptographic algorithms or steps may be introduced. These new features and/or cryptographic steps may require new firmware for the primary platform in order to support new or updated versions of the standards. A need exists in the art for a primary platform to securely receive
35 updated firmware in order to support currently evolving standards and features for the operation of a primary platform or SSP and supported applications.

Secure operation of a primary platform or an SSP for each of the above standards depends on the secure delivery of firmware from a server to the primary platform. Different firmware may be required by a primary platform in order to support each of the above example applications. Further, over time new features may be added to the applications, which would also likely require a firmware update. The GP OFL document provides an overview of a proposed solution in Figure 2-1 on page 15. Details for an OFL security scheme are outlined in the GP OFL document in Figure 4-1 on page 33. Secure encryption and also an authenticated delivery of firmware to a primary platform is a significant technical challenge, since the computing device may include insecure components, such as a generic processor and a generic operating system. The insecure device can conduct many of the steps for communicating between (i) a server that sources the firmware and (ii) the primary platform. As one example, the computing device could comprise a mobile phone or “smartphone” based on Android or IOS or similar operating systems and could also either (i) operate with “malware” that is unknown to a user or a network or (ii) could comprise a “rooted” mobile phone that is under the control of hackers. Many other examples of insecure devices operating a TRE with a primary platform exist as well. Note that the GP OFL document does not include the words “revocation” or “revoked”, which are associated with steps to revoke or invalidate a previously issued certificate from a certificate issuer. A need exists in the art for secure steps for a TRE with a primary platform to determine revocation of a certificate or public key for an image delivery server (IDS), an OFL, or an associated server.

A primary platform or a “Smart Secure Platform” operating in a tamper resistant element can be a resource constrained computing environment, compared to a traditional computing environment of a mobile phone or a personal computer. This resource constrained environment can create significant challenges for revocation of a certificate or public key associated with a server supporting the delivery of firmware to the primary platform. As one example, a primary platform may not have an internal clock to keep track of local date and time when power is removed from the primary platform. The primary platform must communicate through the insecure mobile phone or computing device in order to send and receive data with a server, including updated firmware for the primary platform. The primary platform may not have resources to operate a full transport layer security (TLS) stack. In addition, the primary platform may lack computing resources and electrical components for the proper and full implementation of certificate revocation lists, the Online Certificate Status Protocol (OCSP), or associated schemes for checking the potential revocation of a certificate or public key from a server. A need exists in the art for a primary platform operating in a tamper resistant element to use time values received in order to securely determine if a certificate or public key used with cryptographic operations has been revoked.

Both (i) the long timeframes for a tamper resistant element to record and operate with server certificates, which could comprise certificates structured according to the X.509 family of standards

and include a certificate issuer (CI) root certificate, and (ii) the use of potentially hundreds or more of IDS certificates signed by the CI root private key increases the importance and likelihood of a need in the art for a TRE to support or determine certificate revocation for a certificate received by the TRE. As one example, the root certificate from the GSMA for the system supporting the embedded universal integrated circuit card (eUICC) currently expires in approximately 2052. The CI certificate recorded in devices for a system of TRE as contemplated by GP OFL may similarly expire in future decades. An IDS may communicate with millions of devices with TREs and hundreds of service providers over time, and may prefer to use an IDS certificate that is valid for several years for operational purposes.

The GP OFL contemplates in section 4.2.6 on page 39 checking for validity of an IDS certificate using a timestamp received, but note this only contemplates the verification that an IDS certificate has not expired, where expiration of the IDS certificate may be several years into the future. There is no teaching or suggestion in the GP OFL for an intermediate check or verification by the TRE that if the IDS certificate or OFL authority certificate has been revoked. A need exists in the art for a primary platform in a TRE to conduct a check for certificate revocation where the device operating the TRE may not directly or efficiently or reliably communicate with the certificate issuer (CI) that signed a certificate used by the TRE, such as an IDS certificate or an image owner certificate.

Note that the IDS may be connected to the public Internet in order to support the potential millions of different devices with a TRE requesting firmware for the PP. This creates potential security risks for protecting the private key of the IDS corresponding to the public key stored by the TRE with an IDS certificate, especially over long period of time such as several years. As one example, the “Heartbleed” vulnerability in OpenSSL potentially exposed the private key operated by servers connected to the Internet. The cloud hosting service Cloudflare around 2014 had to revoke and reissue thousands of server certificates for that one vulnerability. Another, similar vulnerability may be found in the future, and several existing threats for server security continue in the families of Spectre and Meltdown vulnerabilities. The list from Common Vulnerabilities and Exposures contains thousands of entries for the year 2018 alone. Many of these vulnerabilities could result in exposure or leaking of bits of security for a private key of a server corresponding to the public key in a certificate operated by a TRE as contemplated by Global Platform and ESTI standards. Consequently, a need exists in the art for a certificate issuer to revoke a certificate used by a primary platform or “Smart Secure Platform” and reissue the certificate in a manner that the TRE can determine that the certificate has been revoked.

Many other examples exist as well for needs in the art for a primary platform to securely receive firmware from a server using certificates where the primary platform can determine if the

certificate has been revoked, and the above are examples are just a few and intended to be illustrative instead of limiting.

SUMMARY

Methods and systems are provided for a primary platform to securely receive firmware from a server. A device can include a tamper resistant element. The primary platform (PP) can operate within the tamper resistant element (TRE) and comprise a secure processing environment operating within a computing device. The primary platform can comprise a secure enclave, a secure element, a trusted execution environment (TEE), or a “Smart Secure Platform” as specified by ETSI standards. The primary platform can also operate within an ARM® Trustzone implementation of a TEE. The TEE can support GlobalPlatform standards for a TEE. The computing device can comprise a mobile phone, wireless device, personal computer, laptop, or a networked transducer for the “Internet of Things”. The device can (i) include insecure components such as a general processor and a general operating system and (ii) communicate with the primary platform using a device driver such as an Open Firmware Loader (OFL) agent.

The device can connect with an internet protocol (IP) based network such as the public Internet in order to establish a secure session with a server. The server can comprise an image delivery server (IDS) and receive a firmware image from an image owner. The server can comprise a computer with a network interface to communicate with the device via the IP network. The server can record and operate a server database. The device can be one of a plurality of different devices communicating with the server. The server can operate an online certificate status protocol (OCSP) client in order to periodically obtain a signed verification of certificate validity from a certificate issuer.

In exemplary embodiments an image owner can obtain a firmware image for the primary platform. The firmware image can support the primary platform operating with an application, such as, but not limited to, an eUICC, a “Smart Secure Platform”, secure identification of the device using the primary platform, and other possibilities exist as well for supported applications. The image owner can obtain the firmware image for a plurality of different primary platforms. The firmware image from the image owner can comprise an unbound image, which could be subsequently used by the server communicating with a plurality of different primary platforms, after subsequent cryptographic operations by the server with the plurality of different primary platforms in order to create bound images. The image owner can receive a signed timestamp from an open firmware loader (OFL) authority, where the OFL authority also provides additional authorization information for the primary platform. The image owner can send the authorization information and the signed timestamp to the server. For some embodiments, the use of a primary platform could be omitted, and the firmware image could comprise firmware for a processor within a device operating in a secure manner (e.g. Trustzone, TEE, “secure enclave”, etc.). In addition, the image owner or IDS

could receive the signed timestamp from a server different than a server associated with the OFL authority.

The primary platform and the server can record and operate a set of compatible cryptographic parameters and cryptographic algorithms for an elliptic curve Diffie Hellman (ECDH) key exchange algorithm, a symmetric ciphering algorithm, and elliptic curve digital signature algorithms (ECDSA). Before distribution to an end user of the computing device, a TRE manufacturer could record a set of data in nonvolatile memory for the TRE. In addition to a PP boot firmware and PP boot configuration for the PP, the data recorded in device before distribution could include a root certificate for a certificate issuer and a certificate for a timestamp authority. Other PKI keys could be recorded by a TRE manufacturer in a set of PKI keys for the primary platform as well

A certificate issuer (CI) can receive a first public key and a second public key from the server and issue certificates for each of the public keys. The certificate issuer can use a first CI private key, corresponding to a CI root certificate recorded by the primary platform, to sign the server public keys. The first public key can be associated with elliptic curve key agreement and the second public key can be associated with elliptic curve digital signature algorithms (ECDSA). The certificate issuer can also (i) operate an OCSP responder to receive queries for interim certificate validity before the expiration time of the certificates and (ii) respond with a signed response and a time value, where the time value represents the end of an interim time into the future that the certificate can be considered valid and not revoked. In exemplary embodiments, the interim time could comprise weekly updates for the time value and interim verification for the validity of the certificate. The responses from the OCSP responder can use a second CI private key, where either (i) a certificate for the second CI private key is also recorded by the primary platform, or (ii) a certificate for the second CI private key is signed by the first CI private key and the certificate is sent along with the response for the interim update for certificate validity.

After boot and startup of both the device and the TRE operating the primary platform, the device can use an OFL agent as a device driver to communicate with the primary platform. The device can use the OFL agent to obtain (i) a firmware status, (ii) the primary platform identity, and (iii) a certificate for the primary platform. The firmware status can comprise a list of parameters associated with any firmware image recorded or stored by the primary platform, including a version number for either a stored firmware image or a version number for PP boot firmware. In exemplary embodiments, the set of cryptographic parameters in the certificate for the primary platform at least specifies a specific ECC curve for use with the primary platform private key. The device can establish a secure session with the server and receive a first server certificate for the server, where the certificate includes OCSP stapling of a current OCSP response from the certificate issuer for the first server certificate. The OCSP response with the first server certificate can include a first time

value. The device can forward the first server certificate with the OCSP response and the first time value to the primary platform using the OFL agent.

The primary platform can receive the first server certificate with the OCSP response and first the time value. The primary platform can use a certificate for the certificate issuer to verify the first time value, where the first time value is stored for a later comparison with a received timestamp. The primary platform can derive an ephemeral PKI key pair comprising a first ephemeral private key and a first ephemeral public key, using parameters from the first server certificate. The primary platform can conduct a first key exchange in order to derive a first symmetric ciphering key and encrypt the certificate for the primary platform using the derived first symmetric ciphering key. The primary platform can send the ephemeral public key and the encrypted certificate for the primary platform to the OFL agent in the device, which forwards the data to the server through the secure connection.

The server can receive the ephemeral public key and the encrypted certificate for the primary platform and conduct a public key validation step for the ephemeral public key. The server can conduct a second key exchange using the received first ephemeral public key and a first server private key corresponding to the server public key sent in the first server certificate, in order to derive the first symmetric ciphering key. The server can decrypt the primary platform certificate using the derived first symmetric ciphering key.

The server can derive an ephemeral PKI key pair which includes a second ephemeral public key and a second ephemeral private key. The server can conduct a series of steps to encrypt the received firmware image and the signed timestamp, where the series of steps include using (i) the public key from the primary platform certificate, (ii) a second private key corresponding to a second public key in a second server certificate, and (iii) the derived second ephemeral private key. The server can send a message to the device with at least the second ephemeral public key, the encrypted image, the signed timestamp, and the second server certificate. The message with the second server certificate can also include OCSP stapling of a current OCSP response from the certificate issuer for the certificate. The OCSP response with the second certificate can include a second time value.

The device can receive the message from the server and use the OFL agent to send the second ephemeral public key, the encrypted image, the signed timestamp, and the second server certificate to the primary platform operating in the tamper resistant element. The primary platform can conduct a series of steps in a firmware load function in order to process the data received. The device can use (a) the received second ephemeral public key and a private key corresponding to the public key in the primary platform certificate to (b) derive at least a second symmetric ciphering key. The primary platform can use the derived symmetric ciphering key and a symmetric ciphering algorithm to decrypt the encrypted data forwarded from the device in the message to the TRE. The primary platform can decrypt the encrypted firmware image. The message can include an access

rights pattern that is signed by the image owner, and the primary platform can verify the signature for the access rights platform using a public key for the image owner. The primary platform can use the access rights pattern to determine if loading and operating with the firmware image is authorized, as well as other constraints for the firmware image.

5 The primary platform can verify the signed timestamp using a recorded certificate for the timestamp authority. The primary platform can use a certificate for the certificate issuer to verify a second time value from the OCSP stapling with the second server certificate. The primary platform can compare both (i) the first time value from OCSP stapling for the first server certificate and (ii) the second time value from OCSP stapling for the second server certificate with (iii) the received,
10 verified timestamp value. If the received, verified timestamp value is before both the first time value and the second time value (where the first time value and the second time value can represent the end time for validity of OCSP stapling values), then the primary platform can continue the firmware load function and establish a firmware session using the received, decrypted, and authenticated firmware from the message.

15 If the received, verified timestamp value is after either the first time value or the second time value, then the primary platform cannot trust or rely that both the first server certificate and the second server certificate remain valid and have not been revoked. If the received, verified timestamp value is after either the first time value or the second time value, then the primary platform can send an error response to the server through the device and stop the firmware load
20 process, so that the received firmware is not loaded until certificates with time values (e.g. outside window of time validity or “expiration” of the OCSP stapling data) greater than the timestamp are received and verified by the primary platform.

BRIEF DESCRIPTION OF THE DRAWINGS

Various exemplary embodiments are described herein with reference to the following drawings, wherein like numerals denote like entities.

5 Figure 1a is a graphical illustration of an exemplary system, where a device with a tamper resistant element and a server securely transfer encrypted firmware from the server to the tamper resistant element, in accordance with exemplary embodiments;

10 Figure 1b is a graphical illustration of hardware, firmware, and software components for a device, including a tamper resistant element with a primary platform, in accordance with exemplary embodiments;

 Figure 1c is a graphical illustration of a primary platform, in accordance with exemplary embodiments;

 Figure 2a is a simplified response from a certificate authority to verify certificate validity, in accordance with exemplary embodiments; and,

15 Figure 2b is a flow chart illustrating exemplary steps for verifying a digital signature using PKI keys, parameters, and data input, in accordance with exemplary embodiments.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS OF THE INVENTION**Figure 1a**

Figure 1a is a graphical illustration of an exemplary system, where a device with a tamper resistant element and a server securely transfer encrypted firmware from the server to the tamper resistant element, in accordance with exemplary embodiments. The system 100 can include a device 102, a server 103, a certificate issuer 104, an image owner 105, an open firmware loader (OFL) authority 107, and a network 114. The various computing nodes depicted in Figure 1a can establish secure sessions such as a Transport Layer Security (TLS) 115 session over an Internet Protocol (IP) network 114, including the TLS 115 session depicted between device 102 and server 103. Server 103 can comprise an image delivery server (IDS). Although a single server 103, device 102, a CI 104, etc. are depicted in Figure 1a, a system 100 can comprise a plurality of the computing nodes. In addition, although a single server 103 is depicted in Figure 1a, the exemplary server 103 shown for system 100 can comprise either different physical computers such as rack-mounted servers, or different logical or virtual servers or instances operating in a “cloud” configuration, including different computing processes which are geographically dispersed.

Device 102 can be a computing device for sending and receiving data, including a wireless device. Device 102 can take several different embodiments, such as a general purpose personal computer, a mobile phone based on the Android ® from Google ® or the IOS operating system from Apple ®, a tablet, a device with a sensor or actuator for the “Internet of Things”, a module for “machine to machine” communications, a device that connects to a wireless or wired Local Area Network (LAN), and other possibilities exist as well without departing from the scope of the present disclosure. Device 102 can also be a computing device according to GSMA technical document “iUICC POC Group Primary Platform requirements”, Approved Release 1.0 dated May 17, 2017, which is hereby incorporated by reference in its entirety (“GSMA PP Requirements”). Device 102 can comprise a device such as that depicted in Figure 6 on page 24 of the GSMA PP Requirements. Exemplary electrical components within a device 102 are depicted and described in Figure 1b below. Other possibilities exist as well for the physical embodiment of device 102 without departing from the scope of the present disclosure.

Server 103 could also represent different logical “server-side” processes within a network, including different programs running on a server that listen and communicate using different IP port numbers within one physical server. In exemplary embodiments, server 103 can operate using the physical electrical components similar to those depicted and described for a device 102 in Figure 1b below. Server 103 can use electrical components with larger capacities and larger overall power consumption, compared to the capacity and power consumption for the equivalent electrical

components in a device 102. Other possibilities exist as well for the physical embodiment of server 103 without departing from the scope of the present disclosure.

IP network 114 could be either a Local Area Network (LAN) or a Wide Area Network (WAN), or potentially a combination of both. IP network 114 could include data links supporting either IEEE 802.11 (WiFi) standards. Device 102 could also utilize a variety of WAN wireless technologies to communicate data in a secure session 115 with server 103, including Low Power Wide Area (LPWA) technology, 3rd Generation Partnership Project (3GPP) technology such as, but not limited to, 3G, 4G Long-Term Evolution (LTE), or 4G LTE Advanced, NarrowBand – Internet of Things (NB-IoT), LTE Cat M, planned future 5G networks, and other examples exist as well. Server 103 can connect to the IP network 114 via a wired connection such as, but not limited to, an Ethernet, a fiber optic, or a Universal Serial Bus (USB) connection (not shown). IP network 114 could also be a public or private network supporting Internet Engineering Task Force (IETF) standards such as, but not limited to, such as, RFC 786 (User Datagram Protocol), RFC 793 (Transmission Control Protocol), and related protocols including IPv6 or IPv4. A public IP network 114 could utilize globally routable IP addresses and also comprise an insecure network.

As depicted in Figure 1a, device 102 can include an OFL agent 102x and a tamper resistant element 113. A device 102 can also include other electrical components and data as depicted and described in connection with Figure 1b below, where the components for a device 102 are described in more detail. An OFL agent 102x can comprise a device program such as a device driver operating on a general processor outside of the tamper resistant element 113. Tamper resistant element (TRE) 113 can comprise a tamper resistant element as described in the GSMA PP Requirements document. Tamper resistant element can comprise a silicon enclave within a tamper resistant chip such as a “system on chip” as depicted and described in connection with Figure 1b below.

TRE 113 in device 102 can include a primary platform (PP) 101, where a primary platform is also described in the GSMA PP Requirements document and the GP OFL document. TRE 113 could also comprise a “Smart Secure Platform” (SSP) as described in ETSI TC SCP Meeting #81 document “SCP(17)000188”, which is hereby incorporated by reference in its entirety. Note that draft specifications for an SSP such as “103 666-1 SSP Draft Specification 0.8.0” are not publicly available and have restricted access on the ETSI web site as of December 15, 2018. Primary platform 101 can comprise a secure operating environment, a secure enclave, a secure element, and include a dedicated processing core within a processor for device 102. Primary platform 101 can also operate in a Trusted Execution Environment (TEE) within a processor for device 102, or also within an ARM® “Trustzone” processing environment. Primary platform 101 can also comprise a SSP as contemplated by ETSI documents and draft specifications for 5G networks.

TRE 113 and PP 101 can support a variety of applications. TRE 113 can comprise the physical device such as that depicted in Figure 1a and Figure 1b below (e.g. silicon, electrical

contacts, housing, etc.), and a primary platform 101 can comprise a secure processing environment operating within the TRE 113 (e.g. operating system, drivers, etc.). With appropriate firmware 106, TRE 113 and PP 101 could operate as an “integrated universal integrated circuit card” (iUICC), an “embedded universal integrated circuit card” (eUICC), a secure element for banking applications or payments from mobile phones, an radio-frequency identity (RFID) card, a secure bootstrap environment for device 102, a virtual key for cars or door locks, an secure environment for recording an identity and secret or private keys for drivers licenses, passports, online or web-site access, etc. Other applications for firmware 106 operating in TRE 113 and PP 101 are possible as well, without departing from the scope of the present disclosure. In general, cryptographic keys and cryptographic algorithms and parameters could be stored in PP 101 in order to securely support applications such as device programs operating on device 102. In this manner, an insecure device program also operating on device 102 would not feasibly be able to ready the cryptographic keys or use the cryptographic algorithms stored in PP 101.

Each of the above exemplary applications can be operated by a firmware 106 running within TRE 113 on PP 101. Although a single firmware 106 is depicted and described in connection with Figure 1a, a TRE 113 and PP 101 can operate a plurality of different firmware 106 simultaneously (or “effectively simultaneously” such that rapid switching of a processor in TRE could make the firmware to appear to operate concurrently over a longer period such as a millisecond, even though only one firmware may operate at one time over a shorter period such as a nanosecond). A primary firmware can comprise the host environment for the primary platform and a secondary firmware can comprise a guest or virtual machine operating in the primary platform. A firmware 106 can comprise either the primary firmware or the secondary firmware as described in the previous sentence and other references herein.

Firmware 106 (as secondary firmware) can operate within a virtual machine for a processor for TRE 113 and PP 101. As one exemplary embodiment, firmware 106 can comprise a JAVA applet running on a java virtual machine in TRE 113, where the JAVA virtual machine can be recorded in secure boot firmware for TRE 113 and PP 101. Different firmware 106 operating within TRE 113 can be isolated from each other by conventional technology for processing hosts and/or virtual machines. Other possibilities exist as well for a TRE 113 and PP 101 to operate as a host for an application downloaded as firmware 106. In summary, the overall security of an application operated by TRE 113 and PP 101 can depend on securely receiving firmware 106 by TRE 113 and PP 101 from a server 103.

PP 101 can include a set of PKI keys 101a, a key exchange algorithm 101b, and a load function 119. Other data, functions, and electrical components can be included in a PP 101, such as the data, functions, and electrical components for a primary platform as described in the GSMA PP requirements document and the GP OFL document. As one example, but not depicted in Figure 1a,

a primary platform 101 can record a globally unique identity for the primary platform and/or TRE 113. A set of PKI keys 101a can record a plurality of certificates and private keys for primary platform 101, which can include a certificate for (i) the IDS server for elliptic curve cryptography key agreement (ECKA) of CERT.IDS1.ECKA 103c and (ii) the chip part number for a elliptic curve digital signature algorithm (ECDSA) of CERT.PN.ECDSA 101a-2. A set of PKI keys 101a can also include the exemplary keys depicted and described in connection with Figure 1c below. CERT.PN.ECDSA 101a-2 along with the corresponding private key can be recorded within PP 101 upon manufacturing of the TRE 113. CERT.IDS.ECKA 103c can be received by PP 101 after manufacturing, such as during setup of a TLS 115 session and before PP 101 conducts the steps for a key exchange algorithm A 101b.

Certificates depicted in Figure 1a such as CERT.IDS.ECKA 101a-1 and CERT.IDS2.ECDSA 103d can specify values or settings for (i) conducting an ECDH or ECDHE key exchange, (ii) mutually deriving a symmetric ciphering key, (iii) an ECC curve which could comprise a commonly used name curve, (iv) PKI key lengths, and/or (iv) an elliptic curve digital signature algorithm (ECDSA), etc. Each of the certificates associated with PP 101, server 103, and other nodes in Figure 1a can record at least one compatible set of cryptographic parameters within the depicted certificates. In exemplary embodiments, the certificates in Figure 1a can specify values for algorithms supporting post-quantum cryptography, such as lattice-based algorithms, code-based algorithms, or multivariate-based algorithms, and other possibilities exist as well without departing from the scope of the present disclosure.

Cryptographic parameters within certificates can specify values for an elliptic curve cryptography (ECC) curve name, key length, key formatting (e.g. compressed or uncompressed), encoding rules, etc. As contemplated herein, the parameters and cryptographic algorithms used with certificates, ECC PKI keys and key exchanges or elliptic curve digital signature algorithms in the present disclosure can be compatible and substantially conform with ECC algorithms and keys as specified in (i) the IETF Request for Comments (RFC) 6090 titled "Fundamental Elliptic Curve Cryptography Algorithms", and (ii) IETF RFC 5915 titled "Elliptic Curve Private Key Structure", and also subsequent and related versions of these standards. In addition, parameters and algorithms for ECDSA in certificates can be compatible and support BSI Technical Guideline TR-03111: Elliptic Curve Cryptography. Version 2.0. Other possibilities exist as well for cryptographic parameters and algorithms specified in a certificate without departing from the scope of the present disclosure.

For use of ECC algorithms, parameters in a certificate can specify elliptic curve names such as, but not limited to NIST P-256, sect283k1, sect283r1, sect409k1, sect409r1, and other possibilities exist as well. Further, elliptic curves that do not depend on curves currently specified by the National Institute of Standards and Technology (NIST) could be utilized as well, such as, but

not limited to, Curve22519, curve448, or FourQ. Parameters in a certificate can specify domain parameters for elements in system 100 to calculate values or numbers in a compatible manner, such as (i) a common base point G for use with ECC PKI key pairs and (ii) a defining equation for an elliptic curve.

5 Key exchange algorithm 101b can use CERT.IDS1.ECKA 103b and a derived ephemeral PKI key pair for PP 101 in order to derive a symmetric ciphering key M1 and message authentication code key H1. The public key for the derived ephemeral PKI key pair can comprise PKE1 depicted for message 117 in Figure 1a. Key exchange algorithm 101b can comprise the steps for section 4.1.2.1 and 4.1.2.2 on page 35 of the GP OFL document. For embodiments that support
10 the use of post-quantum cryptographic algorithms instead of ECC algorithms, the key exchange algorithm 101b can comprise a key exchange mechanism, where the symmetric ciphering key M1 could be encrypted using a public key and decrypted using a private key.

Load function 119 can use data received in message 118, including an ephemeral public key for the IDS server of PKE2 in order to authenticate and decrypt an encrypted firmware 106*. As
15 contemplated herein, an encrypted firmware 106 can be depicted and described herein as “firmware 106*” and a plaintext firmware 106 for a primary platform to establish a firmware session can be depicted and described as “firmware 106”. A load function 119 for a primary platform is also depicted and described in connection with Figure 1c below. Individual steps within a load function 119 in Figure 1a are also described below with the description of message flows between the nodes
20 in system 100 and operation of PP 101.

Firmware 106 recorded in PP 101 upon conclusion of a load function 119 can provide machine executable instructions for a processor in TRE 113 to execute or run. Firmware 106 could comprise a collection of compiled software libraries and programming code for the operation of TRE 113 and PP 101. Firmware 106 could comprise a Java-based applet or application as secondary
25 firmware, where boot firmware of PP 101 establishes and operates a Java virtual machine such as, but not limited to JamVM or HaikuVM. Other platforms for virtualization and emulation of a computer system by PP 101 are possible as well, without departing from the scope of the present disclosure, where firmware 106 can be compiled or formatted to operate on PP 101 operating as a host for the virtualized computer system. In exemplary embodiments, firmware 106 can comprise an
30 application where PP 101 operates as a process virtual machine or an application virtual machine. The environment in which firmware 106 operates can also be referred to as a managed runtime environment (MRE). Firmware 106 could also comprise a primary firmware for the operation of PP 101 as a host computer for the operation of secondary firmware.

Firmware 106 can comprise compiled software or machine executable instructions for either
35 (i) a processor or (ii) a virtual machine in PP 101, and may also be referred to herein as an “image”. In other words, although (A) firmware may traditionally refer to machine executable programming

instructions that provides low-level or hardware control over computing hardware, such as memory and physical interfaces, as contemplated herein, (B) “firmware” can comprise higher level software written for a virtual machine. In addition, the computing environment of a primary platform can require secure functions such as writing and reading cryptographic keys for a firmware 106 specially designated protected memory, and thus firmware 106 comprising high level software may include features similar to traditional firmware. Further, firmware may be traditionally associated with machine executable instructions that are read from a read only memory, and firmware 106 comprising software that is loaded into primary platform 101 can have features after loading in PP 101 that are similar to traditional firmware, such as firmware 106 not being readily modified by an insecure processor in device 102. In any case, although “firmware 106” is described herein as firmware, “firmware 106” can comprise any collection of machine executable instructions which can be loaded and operated by primary platform 101. Similarly, the GSMA PP Requirements document and GP OFL document refer to the collection of machine executable code for a primary platform 101 as “firmware”.

As depicted in Figure 1a, an image delivery server 103 (or “server 103”) can include a public key PK.IDS1.ECKA 103a, a public key PK.IDS2.ECDSA 103b, a record certificate function 103e, an OCSP client 103f, a public key validation function 103g, a key exchange A’ 103h algorithm, a key exchange B’ 103i algorithm, and a OCSP query repeat 103j function. Although not depicted in Figure 1a, a server 103 could record the corresponding private keys for public keys PK.IDS1.ECKA 103a and PK.IDS2.ECDSA 103b. Server 103, PP 101, and the other nodes depicted in Figure 1a can also commonly share cryptographic parameters and cryptographic algorithms as specified in the depicted certificates for Figure 1a and Figure 1c below. The record certificate function 103e for server 103 can record the certificate for both (i) PK.IDS1.ECKA 103a comprising CERT.IDS1.ECKA 103c and (ii) PK.IDS2.ECDSA 103b comprising CERT.IDS2.ECDSA 103d, where the public keys are sent to certificate issuer 104 and the certificates are received from certificate issuer 104.

OCSP client 103f can comprise a client to operating according to the online certificate status protocol (OCSP) according to IETF RFC 6960 titled “X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP”, which is hereby incorporated by reference in its entirety. OCSP client 103f could support future or related standards for periodically verifying or determining if a previously issued certificate has been revoked or is invalid. OCSP client 103f can submit a query 109 for a certificate such as CERT.IDS1.ECKA 103c or CERT.IDS2.ECDSA 103d to CI 104 and receive a response 110 from CI 104. An exemplary query 109 and response 110 for an OCSP client 103f is also depicted and described in connection with Figure 2a below. Note that response 110 from OCSP responder 104d includes a signed time T2 104e, where signed time T2 104e can comprise both a digital signature portion and a plaintext time value for T2 104e-1 (in

Figure 2a). The value for T2 104e-1 can comprise a duration of validity confirmation of CERT.IDS1.ECKA 103c or CERT.IDS2.ECDSA 103d over a shorter period of time than the expiration time of the certificate, such as an exemplary several days or few weeks, instead of the certificate expiration time T1 104c. By receiving the value for T2 104e in an authenticated manner and using the subsequent steps described herein, PP 101 can reasonably be assured that a certificate associated with IDS server 103, such as CERT.IDS1.ECKA 103c or CERT.IDS2.ECDSA 103d has not been revoked.

A key exchange A' 103h algorithm for server 103 can use a received ephemeral ECC public key PKE1 from a message 117 and the secret or private key for PK.IDS.ECKA 103a in order to mutually derive a symmetric ciphering key M1 and MAC key H1 with PP 101. A key exchange A' 103h can include the step "KS13=DERIVE(SK.IDS1.ECKA)[PK.OFL.ECDHE]" in section 4.1.2.3 of the GP OFL document on page 36. Server 103 can then derive an ephemeral PKI key pair that includes an ephemeral public key of PKE2 and a corresponding ephemeral private key. Server 103 can conduct a key exchange B' 103i using the corresponding ephemeral private key and the public key from CERT.PN.ECDSA 101a-2. A key exchange B' 103i can include the step "KS26=DERIVE(SK.IDS2.ECDHE)[PK.OFL.ECDHE]" in section 4.1.2.3 of the GP OFL document on page 36. Although not depicted in Figure 1a, a server 103 can also conduct the other steps from section 4.1.2.3 of the GP OFL document on page 36 along with key exchange A' 103h and key exchange B' 103i in order to prepare and process a message 118 with an encrypted firmware 106* in a secure and authenticated manner.

A server 103 can include an OCSP query repeat function 103j. The OCSP query repeat function 103j can determine if a time value T2 104e from an OCSP response 110 is less than a timestamp T3 108b-1 from an image owner 105. If time value T2 104e is less than timestamp T3 108b-1, then server 103 can repeat the OCSP query 109. Time value T2 104e-1 could be less than timestamp T3 108b-1 for embodiments where T2 104e-1 and timestamp T3 108b-1 are close together and clocks for nodes in a system 100 are not fully synchronized, such as an exemplary minute apart in time. Additional details for the operation of an OCSP query repeat function 103j with associated time values will be described below in the overall sequence of message flows between the nodes.

A system 100 can also include a certificate issuer (CI) 104. CI 04 can comprise a collection of servers to issue certificates and also verify certificate status after issuance. CI 104 can include private or secret keys SK.CI.ECDSA 104a and SK.CI-OCSP.ECDSA 104b, a verify and sign function 104c, certificates CERT.IDS1.ECKA 103c and CERT.IDS2.ECDSA 103d, and an OCSP Responder 104d. Secret key SK.CI.ECDSA 104a can comprise a root private key for the root certificate and public key in CERT.CI.ECDSA 101a-3 recorded in primary platform PKI keys 101a as depicted in Figure 1c below. Secret key SK.CI-OCSP.ECDSA 104b can comprise a private key

corresponding to the public key in CERT.CI- OCSP.ECDSA 101a-4 recorded in primary platform PKI keys 101a as depicted in Figure 1c below. The verify and sign function 104c can receive public keys from IDS 103 such as PK.IDS1.ECKA 103a and PK.IDS2.ECDSA 103b and use the secret key SK.CI.ECDSA 104a in order to generate certificates of CERT.IDS1.ECKA 103c and CERT.IDS2.ECDSA 103d, respectively. Note that CERT.IDS2.ECDSA 103d includes an expiration time T1 104c, which represents the time for end of the validity of the certificate.

CI 104 can operate an OCSP responder 104d. OCSP responder 103d can comprise a responder to operating according to the online certificate status protocol (OCSP) according to IETF RFC 6960 titled “X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP”, which is hereby incorporated by reference in its entirety. OCSP responder 104d could support future or related standards for periodically confirming or reporting if a previously issued certificate has been revoked or is invalid. OCSP responder 104d can receive a query 109 for a certificate such as CERT.IDS1.ECKA 103c or CERT.IDS2.ECDSA 103d from a server such as IDS server 103 and generate a response 110 for the query 109. An exemplary query 109 and response 110 for an OCSP responder 104d is also depicted and described in connection with Figure 2a below.

In exemplary embodiments, CI 104 uses private key SK.CI-OCSP.ECDSA 104b for creating and generating the signed time value T2 104e instead of SK.CI.ECDSA 104a, and in this manner the SK.CI.ECDSA 104a can remain more secured (such as 104a being recorded offline on a different server). Consequently, PP 101 can use CERT.CI- OCSP.ECDSA 101a-4 recorded in PKI keys 101a (shown in Figure 1c) to verify the signed time value T2 104e. In other exemplary embodiments, CI 104 can use the SK.CI.ECDSA 104a to generate a signature for signed time value T2 104e, and in these embodiments TRE 113 and PP 101 can use the CERT.CI.ECDSA 101a-3 (shown in Figure 1c) to validate the signed time value T2 104e using steps equivalent to step 108b in Figure 2b below.

A system 100 can also include an image owner 105. Image owner 105 can comprise a collection of servers to create or record firmware 106 images for PP 101 and also process or issue authentication tokens to enable PP 101 to load and operate firmware 106. The authentication tokens can include information for permission or rights of the TRE 113 to operate with the firmware 106. Note that a TRE manufacturer could also operate as an image owner 105, and other possibilities exist as well for the entity that performs the functionality of an image owner 105. An image owner 105 in a system 100 can be compatible with an image owner as described in the GP OFL document. Although a single firmware 106 image is depicted in Figure 1a, an image owner 105 could record a plurality of firmware 106 images. Image owner 105 can receive signed timestamp T3 108b from an OFL authority 107. Note that a signed timestamp 108b can include both a plaintext time value T3 108b-1 and a digital signature from a timestamp authority (TSA). For some exemplary embodiments, a signed timestamp 108b can include both a plaintext time value T3 108b-1 and a

transaction identity (ID) 117x, where (i) the transaction ID 117x can be generated by TRE 113. Additional details regarding using a signed timestamp T3 108b are provided in Figure 2b below.

A system 100 can also include an OFL authority 107. OFL authority 107 can comprise an entity or collection of servers for granting the rights to an Image Owner for managing an image by using OFL. Although not depicted in Figure 1a, an OFL authority 107 can also send an image owner 105 an authentication token with access rights. In some embodiments, such as when an OFL state becomes “locked”, an image owner 105 can become the OFL Authority 107. As depicted in Figure 1a, and OFL authority can operate with a time stamp authority (TSA) 108. TSA 108 could operate according to the American National Standards Institute standard X9.95 for creating signed timestamps, and other possibilities exist as well. Although depicted as operated by OFL authority 107, TSA 108 could operate outside of OFL authority 107 and be associated with OFL authority 107 or an IDS 103.

TSA 108 can record a secret key SK.TSA 108a corresponding to the public key in CERT.TSA 101a-5 from PP 101 PKI keys 101a in Figure 1c. TSA 108 can operate a clock for tracking date and time that is closely synchronized with established time standards. TSA 108 can create a signed timestamp T3 108b using the SK.TSA 108a. The signed timestamp T3 108b can comprise a digital signature portion and a plaintext timestamp value of T3 108b-1. For some embodiments as depicted in Figure 1a, TSA 108 could include a transaction ID 117x with the signed timestamp T3 108b, where (i) the TSA 108 could receive the transaction ID 117x from a message 117b from IDS 103 and (ii) the IDS 103 could receive the transaction ID 117x from TRE 113 in a message 117. As depicted in Figure 1a, the signed timestamp T3 108b can be received by image owner 105, and image owner 105 can send server 103 a message 112 with both the encrypted firmware 106 and the signed timestamp 108b, along with other data to support an open firmware loader.

The nodes for system 100 depicted in Figure 1a can communicate a series of messages with data in order for PP 101 to receive firmware 106 in a secure and authenticate manner. As described above, IDS server 103 can periodically conduct an OCSP query 109 with CI 104 in order to receive a response 110, where an exemplary response 110 is depicted and described in connection with Figure 2a below. The response 110 can include a digital signature 104e for time value T2 104e-1, where the signed time value T2 104e-1 represents a time where a certificate for IDS 103 such as CERT.IDS1.ECKA 103c or CERT.IDS2.ECDSA 103d have not been revoked. In other words, the signed time value T2 104e-1 can comprise a point in time in the future through which the certificate can be considered valid and not revoked, where the point in time is shorter than the certificate expiration time T1 104c. In exemplary embodiments the signed time value T2 104e-1 could be an exemplary week into the future, while certificate expiration time T1 104c could be several years into the future.

As described above, server 103 could receive firmware 106 image and a digital signature 108b for timestamp value T3 108b-1 from image owner 105 in a message 112. The signed timestamp value T3 108b-1 could originate from a TSA 108. Note that IDS 103 could receive message 112 before or after device 102 and IDS server 103 establish a secure session such as TLS 115 (for embodiments where a transaction ID 117x is not included in digital signature 108b for at least T3 108b-1). For embodiments where a transaction ID 117x is included in digital signature 108b, then message 112 could be received by IDS 103 after (i) the receipt of a message 117 from TRE 113 and (ii) sending or transmitting a transaction ID value 117x to TSA 108 in a message 117b.

Although timestamp value T3 108b-1 (originating from a timestamp authority 108) along with signature 108b is depicted in Figure 1a as being received by server 103 from an image owner 105, timestamp value T3 108b-1 along with signature 108b could be received from another entity for a system 100 besides image owner 105. For some embodiments, server 103 could operate as a timestamp authority and generate timestamp value T3 108b-1 along with signature 108b. Device 102 and server 103 can establish a secure session, which could comprise a TLS session 115. For system 100, server 103 and device 102 may establish a secure session 115, which could comprise establishing a secure communications link between the two servers using protocols such as TLS, IPSec, a virtual private network (VPN), a secure shell (SSH), or similar networking, transport, or application layer technologies in order to establish secure communications between server 103 and device 102.

Secure session 115 is depicted in Figure 1a as “TLS 115” but other methods for establishing secure sessions could be used as well, including Datagram Transport Layer Security (DTLS). Secure session 115 can utilize certificates for server 103 and/or device 102 in order to provide mutual authentication and mutual key derivation for a symmetric encryption key in secure session 115. For some embodiments, the use of a certificate for device 102 in secure session 115 could be omitted, and for these embodiments only server 103 would be authenticated. Other possibilities exist as well for establishing a secure session 108 between server 103 and device 102 without departing from the scope of the present disclosure. Server 103 can begin listening for incoming messages from a device 102 using a physical network interface that provides connectivity to the IP network 114 and server 103 can use a specific port number such as TCP port 443 to listen for incoming data for secure session 115 from a device 102. Note that TLS session 115 is established between device 102 and server 103, and a TRE 113 may not establish the secure session directly, but rather send and receive data over a secure session established by device 102.

As depicted in Figure 1a, server 103 can use an OCSP Query Repeat 103j step after the receipt or generation of a signed timestamp T3 108b-1, such as the receipt in a message 112. For embodiments where (i) a clock to generate T3 108b-1 and (ii) the validity time set by a CI 104 and specified in T2 104e-1 are close together, such as a few seconds apart, there could be instances

where time for received T3 108b is after the time value T2 104e-1. Server 103 could detect that T3 108b-1 is after T2 104e-1 in an OCSF Query Repeat 103j step before sending message 118 and consequently “refresh” or repeat the query 109 with responder 101d in order to obtain a response 110 with a signed time value T2 104e-1 that is after the signed timestamp T3 108b-1.

5 Server 103 can then send device 102 a message 116, where message 116 can include a value for code M and the certificate for server 103 comprising CERT.IDS1.ECKA 103c. Code M can comprise a challenge or nonce from server 103 which will subsequently be signed and used by PP 101 in order to authenticate PP 101 with server 103. Note that in exemplary embodiments, CERT.IDS1.ECKA 103c in message 116 can utilize OCSF stapling, which is also known as the TLS
10 Certificate Status Request extension. The extension is described in section 8, “Certificate Status Request” of IETF RFC 6066 titled “Transport Layer Security (TLS) Extensions: Extension Definitions”. In other words, the certificate 103c in message 116 can include a response 110 from OCSF responder 104d, where a response 110 is also depicted and described in connection with Figure 2a below. An extension for certificate 103c verifying CERT.IDS1.ECKA 103c has not been
15 revoked for an interim period could utilize related or future versions of the standards as well.

Note that CERT.IDS1.ECKA 103c in message 116 can include a first signed time value T2 104e-1 where the digital signature comprises 104e. A second time value T2 104e-1 for CERT.IDS2.ECDSA 103d can be included in message 118 below. In exemplary embodiments, message 116 could also include a CERT.CI- OCSF.ECDSA 101a-4 (not shown in Figure 1a but
20 shown in Figure 1c), where CERT.CI- OCSF.ECDSA 101a-4 can be used to verify the first signed time value T2 104e-1 in the form of digital signature 104e. For embodiments where CERT.CI- OCSF.ECDSA 101a-4 is included in a message 116, then PP 101 can use certificate CERT.CI.ECDSA 101a-3 to validate the certificate CERT.CI- OCSF.ECDSA 101a-4.

Device 102 can receive message 116 and forward data in the message to PP 101 through
25 OFL agent 102x. PP 101 can receive the data from message 116 and conduct steps in section 4.1.2.1 and 4.1.2.2 of the GP OFL document, including a key exchange 101b in order to encrypt CERT.PN.ECDSA 101a-2. PP 101 can then respond to OFL agent 102x with a derived ephemeral public key PKE1 and the encrypted CERT.PN.ECDSA 101a-2. Device 102 using a TRE 113 can also generate a transaction ID 117x, where transaction ID 117x can comprise a pseudo-random
30 number or other unique number to uniquely identify a subsequent message 117 and response 118. Device 102 can then send server 103 a message 117, where message 107 includes the derived ephemeral public key PKE1 and the encrypted CERT.PN.ECDSA 101a-2, and also the transaction ID 117x. Note that message 117 can include other data from steps in section 4.1.2.1 and 4.1.2.2 of the GP OFL document.

35 Server 103 can receive message 117 and conduct the series of steps to verify the received ephemeral public key PKE1 and derive a symmetric ciphering key that was used to encrypt

CERT.PN.ECDSA 101a-2. Server 103 can conduct a public key validation step 103g on received ephemeral public key PKE1 in order to ensure the key is valid and on the selected curve in parameters for CERT.IDS1.ECKA 103a. Step 103g by server 103 can comprise conducting the steps for an ECC Full Public-Key Validation Routine in section 5.6.2.3.2 of FIPS publication SP 800-56A (revision 2) for the received device ephemeral public key PKE1. Alternatively, step 103g can comprise server 103 performing the steps ECC Partial Public-Key Validation Routine in section 5.6.2.3.3 of the same FIPS publication. Other example steps within a public key validation step 103g can comprise (i) verifying the public key is not at the “point of infinity”, and (ii) verifying the coordinates of the point for the public key are in the range $[0, p-1]$, where p is the prime defining the finite field. Other possibilities exist as well for evaluating and validating a received public key is cryptographically secure in a public key validation step 103g, without departing from the scope of the present disclosure.

In exemplary embodiments, a public key validation step 103g can include verifying that a point or public key received in a message such as message 117 is not reused, and in this manner a step 103g can be used to reject messages 299 that could comprise a “replay attack”. Server 101 could record in a server database a list of received PP 101 ephemeral public keys PKE1 for an extended period of time, and if any key PKE1 is reused then message 117 could be rejected. The period of time could be suitable for the application used by TRE 113 and server 103, including potentially a period of years. As contemplated in the present disclosure a PP 101 and server 102 can conduct a public key validation step 103g any time a public key or a point on an elliptic curve is received.

Server 103 can conduct the key exchange A' 103h with the PKE1 and secret key for PK.IDS1.ECKA 103a in order to derive the symmetric ciphering key. Server 103 can use the symmetric ciphering key to decrypt CERT.PN.ECDSA 101a-2 and read and process data using the plaintext certificate. Note that transaction ID 117a can be included either (i) within ciphertext that includes CERT.PN.ECDSA 101a-2 or (ii) as plaintext within message 117 and external to the ciphertext. For some exemplary embodiments as depicted in Figure 1a, server 103 can then send the Timestamp Authority (TSA) 108 the transaction ID 117a in a message 117b.

Server 103 can use the steps in section 4.1.2.3 of the GP OFL document in order to process data for a message 118 with the ciphertext firmware 106*. Server 103 can receive the data for a message 118 to send to device 102 from image owner 105 in a message 112. Message 118 can include both (i) a digital signature 108b for timestamp T3 108b-1 and (ii) the timestamp T3 108b-1. A server 103 can conduct the key exchange step B' 103i when conducting the steps for section 4.1.2.3 of the GP OFL document. Server 103 can then send device 102 a message 118, where message 118 includes an ephemeral public key PKE2, (i) the ciphertext firmware 106* (ii) an encrypted CERT.IDS2.ECDSA 103d, and (iii) a timestamp T3 108b-1 along with the digital

signature 108b for the timestamp T3 108b-1. For some exemplary embodiments, both the timestamp T3 108b-1 and the digital signature 108b for the timestamp T3 108b-1 can be included within encrypted data within message 118, such as within a value M4 from the load function 119 depicted in Figure 1c below.

5 Note that in exemplary embodiments, CERT.IDS2.ECDSA 103d in message 118 can utilize OCSP stapling, which is also known as the TLS Certificate Status Request extension as described above. In other words, the certificate 103d in message 118 can include a response 110 from OCSP responder 104d, where a response 110 is also depicted and described in connection with Figure 2a below. An extension for certificate 103d verifying CERT.IDS2.ECDSA 103d has not been revoked
10 for an interim period could utilize related or future versions of the standards as well. Note that CERT.IDS2.ECDSA 103d in message 118 can include a second time value T2 104e. A first time value T2 104e for CERT.IDS1.ECKA 103c could be included in message 116 above. The timestamp T3 108b-1 can be from TSA 108 and received by server 103 in message 112.

15 In exemplary embodiments, the timestamp T3 108b can be different than the “timestamp” generated by IDS1 in Figure 4.1 of the GP OFL document. The timestamp generated by IDS1 in the GP OFL document (i) is not signed using a digital signature algorithm such as ECDSA and (ii) verified by TRE 113 using a certificate such as Cert.TSA 101a-5 (depicted in Figure 1c below). In other words, the GP OFL document does not suggest a separate timestamp or verification means other than the generation of a timestamp and encryption with a static public key from an image
20 delivery server. Further, there is no teaching or suggestion in the GP OFL document for using the transaction ID to generate the timestamp depicted in Figure 4.1 of the GP OFL document.

Device 102 can receive message 118 and forward PP 101 in TRE 113 data from the message using OFL agent 102x. PP 101 can receive data in message 118. PP 101 can then conduct a load function 119, where details for a load function 119 are also depicted and described in connection
25 with Figure 1c below. In summary, load function 119 for PP 101 can conduct the series of steps for section 4.1.2.4 of the GP OFL document on page 37 with at least the additional steps depicted in Figure 1a. Note that steps in section 4.1.2.4 of the GP OFL document on page 37 include the steps to derive symmetric ciphering keys and convert the encrypted data in message 118 into plaintext. As depicted in Figure 1a, the use of brackets such as “{ }” depicts that the data in a message is
30 encrypted with a symmetric ciphering key and a symmetric ciphering algorithm.

In a step 119a, PP 101 can verify the digital signature for both (i) the first time value T2 104e for CERT.IDS1.ECKA 103c (in OCSP stapling extensions for the certificate) and (ii) the second time value T2 104e for CERT.IDS2.ECDSA 103d (again in OCSP stapling extensions for the certificate) using CERT.CI- OCSP.ECDSA 101a-4 within PKI keys 101a as depicted in Figure
35 1c. Note that the first time value T2 104e-1 can be received by PP 101 along with the digital signature 104e in a message 116 and the second time value T2 104e-1 can be received by PP 101

along with the digital signature 104e in a message 118. The exemplary logic for a step 119a is also depicted for a load function 119 in Figure 1c below, such as the depicted step of “VERIFY(CERT 101a-4) [Signed T2 104e]”. An exemplary signature verification step using a public key, a value, and a signature is also depicted and described in connection with Figure 2b below. As described above, T2 104e can comprise two portions of a signature and the plaintext time value T2 104e-1, as depicted and described in connection with Figure 2a below. Upon conclusion of a step 119a, PP 101 can record the first and second plaintext time values T2 104e-1 for use in a subsequent time comparison step 119c (where the first plaintext time value T2 104e-1 was with CERT.IDS1.ECKA 103c and the second plaintext time value T2 104e-1 was with the CERT.IDS2.ECDSA 103d).

Note that the verification step 119a confirms the signatures 104e for the first time value T2 104e-1 and second time value T2 104e are valid, but PP 101 may need to verify and read a plaintext value for timestamp T3 108b-1 before determining if the first time value T2 104e-1 and second time value T2 104e are acceptable time values (such as determining if time T3 108b-1 is in the future of a time T2 104e, which could be an error or unacceptable state since a certificate may have been revoked after time T2 104e and before time T3 108b). In other words, since PP 101 may not have the functionality to keep local or UTC date and time, additional data and steps can be required in order to determine if CERT.IDS1.ECKA 103c and CERT.IDS2.ECDSA 103d have not been revoked or remain reasonably valid using both (i) first time T2 104e and second time T2 104e, and (ii) time T3 108b.

A step 119b within load function 119 can comprise PP 101 verifying a signature 108b for timestamp T3 108b-1 using a public key from a certificate associated with TSA 108, such as CERT.TSA 101a-5 in PP 101 PKI keys 101a as depicted in Figure 1c below. The exemplary logic for a second step 119a is also depicted for a load function 119 in Figure 1c below, such as the depicted step of “Verify(CERT 101a-5) [Signed T3 108b]”. Step 119b can also comprise PP 101 storing the timestamp T3 108b-1 for use in subsequent steps. Note that the signature 108b for timestamp T3 108b-1 can also be over the transaction ID 117x generated by TRE 113 and PP 101 in the processing of a message 117. Part of a step 119b can include verifying that digital signature for timestamp T3 108b is also over (or includes) the transaction ID 117x. In this manner, TRE 113 and PP 101 can trust that the signature 108b for timestamp T3 108b-1 is current and generated after TRE 113 creates transaction ID 117x and sends message 117. In other words, the use of transaction ID 117x helps prevent a “reply attack” by server 103 simply sending an old timestamp value.

A step 119c within load function 119 can comprise PP 101 comparing the first and second plaintext time values T2 104e-1 and timestamp T3 108b-1. Again, PP 101 may not keep date and time due to limited resources of a TRE 113, and thus can rely on time values with digital signatures that have been previously verified in steps 119a and 119b in order to determine if a certificate remains valid or reasonably determine that a certificate has not been revoked. In step 119c, PP 101

can compare timestamp T3 108b-1 with both the first time T2 104e-1 associated with CERT.IDS1.ECKA 103c (from message 116) and the second time T2 104e-1 associated with CERT.IDS2.ECDSA 103d (from message 118). The time values T2 can represent the “next update” date and time from an OCSP response 110 as depicted in Figure 2a, for both of the certificates 103c and 103d. The status of the certificates in an OCSP response can also be “good” or the equivalent of not being revoked. Time values T2 should be in the future when representing the “Next Update” time within an OCSP response 110 as depicted in Figure 2a (while PP 101 may not track local or global time).

PP 101 can compare the first and second time values T2 104e-1 (from message 116 and message 118) with timestamp T3 108b-1 in a step 119c. If the time values T2 104e-1 are greater than timestamp T3 108b, then T3 108b can be considered in the future and certificates CERT.IDS1.ECKA 103c and CERT.IDS2.ECDSA 103d can be accepted and PP 101 can proceed to step 119d for establishing a firmware session with plaintext firmware 106. Alternatively, PP 101 could determine that timestamp T3 108b is less than either of time values T2 104e-1 in a step 119c in order to accept the certificates 103c and 103d. Although not depicted in Figure 1a, TRE 113 and PP 101 could also use the “timestamp” value from the GP OFL document in Figure 4.1 (“GP timestamp”) to additionally confirm proper time values. For example, the PP 101 could confirm that the GP timestamp is less than the first and second time values T2 104e for certificates 103c and 103d, respectively.

Note that PP 101 should perform the other steps for a load function 119 such as certificate verification, at least one ECDH key exchange to derive a symmetric key, and symmetric decryption of ciphertext in message 118 in order to read an authenticated the plaintext firmware 106. Exemplary additional steps are depicted and described in connection with Figure 1c, and also the exemplary additional steps could comprise the steps within section 4.1.2.4 of the GP OFL document. Note that for some embodiments, the use of two separate time values T2 104e-1 and/or two separate certificates such as CERT.IDS1.ECKA 103c (from message 116) and CERT.IDS2.ECDSA 103d (from message 118) could be omitted, and the use of a single time value T2 104e-1 for a single certificate could be use in a step 119c.

For a first portion of step 119c, if timestamp T3 108b-1 is either in the future or greater than the first time value T2 104e-1 (e.g. from a message 116), then PP 101 cannot be reasonably assured or confirm that CERT.IDS1.ECKA 103c remains valid and the certificate could potentially be revoked. An equivalent determination in a step 119c can be if the first time value T2 104e-1 is less than or at a past time compared to timestamp T3 108b-1. For this case PP 101 can stop the firmware 106 load function 119 and conduct other steps or queries with server 103 in order to notify or attempt to rectify the error state. In other words, PP 101 may not be able to confirm CERT.IDS1.ECKA 103c remains valid at the time of timestamp T3 108b-1. Alternatively PP 101

could determine that the first time value T2 104e-1 is in the future or greater than timestamp T3 108b-1, and thus CERT.IDS1.ECKA 103c was reasonably valid and not revoked at the time timestamp T3 108b-1 was created. If the first time value T2 104e-1 is in the future or greater than timestamp T3 108b-1, then PP 101 can proceed with a second portion of step 119c before loading
5 firmware 106 and step 119d for creating a firmware session to begin operating with the firmware 106 (after the next check of the second time value T2 104e-1 in the next paragraph).

For a second portion of step 119c, if timestamp T3 108b-1 is in the future or greater than or after a second time value T2 104e-1 associated with certificate 103d, then PP 101 cannot be reasonably assured or confirm that CERT.IDS2.ECDSA 103d has been revoked. An equivalent
10 determination in a step 119c can be if the second time value T2 104e-1 is less than or at a past time compared to timestamp T3 108b. For this case PP 101 can stop the firmware 106 load function 119 and conduct other steps or queries with server 103 in order to notify or attempt to rectify the error state. In other words, PP 101 cannot confirm CERT.IDS2.ECDSA 103d remains valid at the time of timestamp T3 108b-1.

Alternatively, for a second portion of step 119c, PP 101 could determine that the second time
15 value T2 104e-1 is in the future or greater or after timestamp T3 108b-1, and thus CERT.IDS2.ECDSA 103d was reasonably valid and not revoked at the time timestamp T3 108b-1 was created. If the second time value T2 104e-1 is in the future or greater than or after timestamp T3 108b-1 (and the first portion of step 119c is also successful), then PP 101 can proceed with
20 loading firmware 106 and step 119d for creating a firmware session to begin operating with the firmware 106. The conclusion of a step 119d for PP 101 can comprise the exemplary state of loaded firmware in state 3 of loaded firmware in Figure 2-5 of section 2.4 "Loading Procedure" of the GP OLF document. Other examples exist as well for (i) loading firmware 106 after a step 119c to compare time values and (ii) establishing a firmware session in a PP 101 without departing from the
25 scope of the present disclosure.

Figure 1b

Figure 1b is a graphical illustration of hardware, firmware, and software components for a device, including a tamper resistant element with a primary platform, in accordance with exemplary
30 embodiments. Figure 1b is illustrated to include many components that can be common within a device 102, and device 102 may also operate in a wireless configuration in order to connect with a wireless network. In a wireless configuration, the physical interface 102a of device 102 may support radio-frequency (RF) communications with networks including a wireless network via standards such as GSM, UMTS, mobile WiMax, CDMA, LTE, LTE Advanced, 5G, and/or other mobile-
35 network technologies. In a wireless configuration, the physical interface 102a may also provide

connectivity to local networks such as 802.11 WLAN, Bluetooth, Zigbee, or an IEEE 802.15.4 network, among other possibilities. In a wireless configuration, device 102 could use a physical interface 102a connected with both a wireless WAN and wireless LAN simultaneously. In a wired configuration, the physical interface 102a can provide connectivity to a wired network such as through an Ethernet connection or USB connection.

The physical interface 102a can include associated hardware to provide connections to components such as radio-frequency (RF) chipsets, an RF amplifier, an antenna, cable connectors, RF filters, etc. Device drivers 102g can communicate with the physical interfaces 102a, providing hardware access to higher-level functions on device 102. Device drivers 102g may also be embedded into hardware or combined with the physical interfaces. Device drivers 102g can include an OFL agent 102x, which can be utilized by a device 102 and operating system 102h in order to read and write data to TRE 113, including communicating with a primary platform 101 within TRE 113. Device 102 may preferably include an operating system 102h to manage device drivers 102g and hardware resources within device 102. The operating systems described herein can also manage other resources such as memory and may support multiple software programs or software libraries operating on device 102, including applications that communicate with PP 101 through a device driver 102g.

The operating system 102h can include Internet protocol stacks such as a User Datagram Protocol (UDP) stack, Transmission Control Protocol (TCP) stack, a domain name system (DNS) stack, etc., and the operating system 102h may include timers and schedulers for managing the access of software to hardware resources, including TRE 113. The operating system shown of 102h can be appropriate for a low-power device with limited memory and CPU resources (compared to a server 103). Example operating systems 102h for a device 102 includes Linux, Android® from Google®, iOS from Apple®, Windows® Mobile, or Open AT® from Sierra Wireless®. Additional example operating systems 102h for device 102 include eCos, uC/OS, LiteOs, Contiki, OpenWRT, Raspbian, and other possibilities exist as well without departing from the scope of the present disclosure.

A device program 102i may be an application programmed in a language such as, but not limited to, C, C++, Java, and/or Python, and could provide functionality to support M2M applications such as remote monitoring of sensors and remote activation of actuators. A device program 102i could also comprise an application for a mobile phone, table, personal computer, or the like. Device program 102i could also be a software routine, subroutine, linked library, or software device, according to one preferred embodiment. As contemplated herein, a device program 102i may be an application operating within a smartphone, such as an iPhone® or Android®-based smartphone, and in this case device 102 could comprise the smartphone. The application functioning as a device program 102i could be downloaded from an “app store” associated with the

smartphone. Device program 102i can include secure session library 102y, which can provide the functionality or “System on a Chip” (SOC) 109 instructions for conducting secure session 115.

SOC 109 can include a plurality of processing cores appropriate for typically low power consumption requirements for a device 102 (compared to a server 103), and may also function as a microcontroller. The processing cores can include at least one processor core for device 102 such as an ARM® based processor or an Intel® based processor such as belonging to the Atom or MIPS family of processors, and other possibilities exist as well including exemplary ARM Cortex-M processors and cores. A single or set of processor or processing cores could operate as a processor for device 102 and a second processor could operate as a TRE 113. Processing cores within SOC 109 can also comprise specialized processing cores, such as including hardware based cryptographic accelerators, field-programmable gate arrays (FPGAs), and other possibilities exist as well without departing from the scope of the present disclosure. For some embodiments, one core could be “split” or divided via time multiplexing such that during a first time period the processing core operates as a generic processor for device 102, and during a second time period the processing core operates as a primary platform 101. The embodiment from the previous sentence could comprise using an ARM “Trustzone” implementation for the TRE 113 and PP 101 for the system 100 and other systems herein.

Many of the logical steps for operation of device 102 can be performed in software and hardware by various combinations of sensor 102f, actuator 102z, physical interface 102a, device driver 102g, operating system 102h, device program 102i, and SOC 109. Note that device 102 may also optionally include user interface 102j which may include one or more devices for receiving inputs and/or one or more devices for conveying outputs. User interfaces are known in the art for devices 102 and could include a few LED lights or LCD display or OLED display, and thus user interfaces are not described in detail here. User interface 102j could comprise a touch screen if device 102 operates as a smartphone or mobile phone. As illustrated in Figure 1b, device 102 can optionally omit a user interface 102j, since no user input may be required for many “Internet of Things” applications, although a user interface 102j could be included with device 102.

Device 102 may be a computing device or wireless device that includes computer components for the purposes of collecting data from a sensor 102f or triggering an action by an actuator 102y. Device 102 may include a central processing unit (CPU) within SOC 109, a random access memory (RAM) 102e, and a system bus 102d that couples various system components including the random access memory 102e to the processing unit 102b. The system bus 102d may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures including a data bus.

Device 102 may include a read-only memory (ROM) 102c which can contain a boot loader program. Although ROM 102c is illustrated as “read-only memory”, ROM 102c could comprise

long-term memory storage chipsets or physical units that are designed primarily for writing once and reading many times, such as Electrically Erasable Programmable Read-Only Memory (EEPROM). As contemplated within the present disclosure, a read-only address could comprise an address within a read only memory such as a ROM 102c memory address or another hardware address for read-only operations accessible via bus 102d. Changing data recorded in a ROM 102c can require a technician have physical access to device 102, such as removing a cover or part of an enclosure, where the technician can subsequently connect equipment to a circuit board in device 102, including replacing ROM 102c. ROM 102c could also comprise a nonvolatile memory, such that data is stored within ROM 102c even if no electrical power is provided to ROM 102c.

Device 102 can include a SOC 109. SOC 109 can include TRE 113, and additional details for the operation of SOC 109 and TRE 113 is provided in other figures herein. Although TRE 113 is depicted in Figure 1b as operating within SOC 109, TRE 113 could be operated within a removable unit such as an SD card, a SIM card, etc. Or TRE 113 could operate within a separate soldered chip connected to bus 102d. An exemplary removable form factor for TRE 113 could comprise a standard SD card, a mini SD card, a micro SD card, a mini UICC, a micro UICC, or a nano UICC, and other possibilities exist as well without departing from the scope of the present disclosure. SOC 109 can include electrical contacts (such as external pads 109a depicted in Figure 1c) which provide electrical connectivity to bus 102d. SOC 109 can include NAND or NOR flash memory in order to record data when device 102 is not powered, and other nonvolatile memory technologies can be used in a storage unit as well without departing from the scope of the present disclosure.

The inclusion of TRE 113 and the operation of TRE 113 with PP 101 in SOC 109 can add functionality for SOC 109 that is not normally included in commercially available SOC in the market as of 2018, such as with the secure receipt of firmware 106 as described herein. TRE 113 within SOC 109 can include a processor, bus, and memory similar (but with less power and on a smaller scale) equivalent to bus 102d, and ROM 102c, NVM 102s. TRE 113 can perform cryptographic functions using either boot firmware or downloaded firmware 106 such as (i) internally deriving a private key with a corresponding public key such as PKE1 (in message 117 in Figure 1a) in a cryptographically secure manner, (ii) recording the private key in a protected memory such that device 102 or external parties cannot feasibly or cost-effectively read the derived private key, and (ii) conducting key exchanges and encryption/decryption such as key exchange 101b from Figure 1a.

Although the exemplary environment described herein employs ROM 102c, RAM 102e, and nonvolatile memory (NVM) 102s, it should be appreciated by those skilled in the art that TRE 113 could also operate within other types of computer readable media which can store data that is accessible by a device 102, such as memory cards, subscriber identity device (SIM) cards, local miniaturized hard disks, and the like, which may also be used in the exemplary operating

environment without departing from the scope of the disclosure. The memory and associated hardware illustrated in Figure 1b provide nonvolatile storage of computer-executable instructions, data structures, program devices, device program 102i, device drivers 102g, and other data for computer or device 102. Note the device 102 may include a physical data connection at the physical interface 102a such as a miniaturized universal serial bus adapter, firewire, optical, or other another port and the computer executable instructions such as device program 102i, operating system 102h, or device driver 102g can be initially loaded into memory such as ROM 102c or NVM 102s through the physical interface 102a before device 102 is given to an end user, shipped by a manufacturer to a distribution channel, or installed by a technician.

Further, device program 102i, operating system 102h, or device driver 102g can be separately loaded into NVM 102s before or after distribution of device 102. In some exemplary embodiments, applications or programs operating within device 102 can be given limited or restricted access to TRE 113 in order to support the applications or programs. For example, a mobile payment application operating a device program 102i could authenticate either device 102 or a user with keys recorded in TRE 113 and a firmware 106. Device program 102i could provide a graphical user interface (GUI) to a user through user interface 101j. Other possibilities exist as well for a device program 102i to operate in conjunction with keys and identities recorded in TRE 113 without departing from the scope of the present disclosure.

A number of program devices may be stored in RAM 102e, ROM 102c, or NVM 102s, including an operating system 102h, device driver 102g, an http client (not shown), a DNS client, and related software. TRE 113 can record program devices as well, where the program devices in TRE 113 may be focused on cryptographic operations and functions conducted within TRE 113 in support of the operation of device 102. A firmware 106 depicted and described in connection with Figure 1a and other figures herein can comprise a program device. Program devices include routines, sub-routines, programs, objects, components, data structures, etc., which perform particular tasks or implement particular abstract data types. Aspects of the present disclosure may be implemented in the form of (i) a device program 102i which are executed by the device 102 working in conjunction with (ii) firmware 106 on TRE 113 and PP 101 to authenticate device 102 with a server using public key infrastructure. In exemplary embodiments, program devices for TRE 113 in SOC 109 can include cryptographic algorithms to operate with the cryptographic parameters specified by certificates recorded in the nodes of system 100, such as with PKI keys 101a.

A user may enter commands and information into device 102 through an optional user interface 102j, such as a keypad, keyboard (possibly miniaturized for a mobile phone form-factor), and a pointing device. Pointing devices may include a trackball, an electronic pen, or a touch screen. A user interface 102j may also include a display (not shown) such as a device screen. A display may also be connected to system bus 102d via an interface. The display can comprise any type of display

devices such as a liquid crystal display (LCD), a plasma display, and an organic light-emitting diode (OLED) display. Device 102 may also include a camera (not shown) connected to or integrated with device 102 through a physical interface 102a, and the camera can comprise a video camera for the device 102 to collect sensor data that includes video or images. The camera (not shown) can be a
5 CCD (charge-coupled device) camera, a CMOS (complementary metal-oxide-semiconductor) camera, or a similar device to collect video or camera input including QR codes. Other arrangements could be used as well, without departing from the disclosure.

The device 102, comprising a computer, may operate in a networked environment using logical connections to one or more remote computers, such as servers. Servers communicating with
10 device 102 can also function as a general purpose server to provide files, programs, disk storage, remote memory, and other resources to device 102 usually through a networked connection. Additional remote computers with which device 102 communicates may include another device 102 or mobile device, an M2M node within a capillary network, a personal computer, other servers, a client, a router, a network PC, a peer device, a wireless network, or other common network nodes.
15 The servers or networks communicating with device 102 or a remote computer typically includes many of the elements and electrical components described above relative to the device 102, including a CPU, memory, and physical interfaces. It will be appreciated that the network connections shown throughout the present disclosure are exemplary and other means of establishing a wireless or wired communications link may be used between mobile devices, computers, servers,
20 corresponding nodes, and similar computers. The operation of a TRE 113 within device 102 with a firmware 106 can be utilized to authenticate a device 102 in each or any of the above described networking environments.

The device program 102i operating within device 102 illustrated in Figure 1b and communicating with TRE 113 can provide computer executable instructions to hardware such as
25 SOC 109 through a system bus 102d in order for a device 102 to (i) transmit and receive data with a service provider, (ii) monitor a sensor and/or change the state of an actuator 102y, (iii) send or receive packets with a server or network, and (iv) authenticate with a server, thus allowing the server to remotely monitor or control device 102 in an authenticated and secure manner. The device program 102i can enable the device 102 to authenticate and communicate with a server by recording
30 data in memory such as RAM 102e, where the data can include sensor data, a destination IP address number, a packet or packet header value, an encryption or ciphering algorithm and key, a digital signature and public key, etc, where cryptographic operations or calculations for the device program 102i can be performed by TRE 113 using firmware 106. The data recorded in RAM 102e can be subsequently read by the operating system 102h or the device driver 102g. The operating system
35 102h or the device driver 102g can write the data to a physical interface 102a using a system bus 102d in order to use a physical interface 102a to send data such as a digital signature for

authentication to a server using the Internet 114. In exemplary embodiments, the digital signature can be generated or processed in the TRE 113 using a PP 101 and firmware 106. Alternatively, the device program 102i can write the data directly to the physical interface 102a using the system bus 102d.

5 In general, digital signatures for authentication with a server can be performed in TRE 113, where the digital signature output is transferred from TRE 113 to RAM 102e using a device driver 102g before being transmitted from device 102 to a server through the IP network 114. The data recorded in RAM 102e such as a digital signature can be subsequently read by the operating system 102h or the device driver 102g. Note that device driver 102g can include OFL agent 102x in order
10 to communicate with TRE 113. Thus, OFL agent 102x can be a device driver 102g specifically for TRE 113. The operating system 102h or the device driver 102g can write the data to a physical interface 102a using a system bus 102d in order to use a physical interface 102a to send data such as a digital signature for authentication to a server using the Internet 114. Alternatively, the device program 102i can write the data directly to the physical interface 102a using the system bus 102d.
15 Other possibilities exist as well without departing from the scope of the present disclosure.

The device program 102i or operating system 102h using TRE 113 and PP 101 with firmware 106 can include steps to process the data recorded in memory such as encrypting data, selecting a destination address, or encoding sensor data acquired by (i) a sensor 102f or (ii) through a physical interface 102a such as a thermocouple, shock or vibration sensor, light sensor, or global
20 positioning system (GPS) receiver, etc. The device 102 can use the physical interface 102a such as a radio to transmit or send (i) the data from a sensor or (ii) a digital signature from TRE 113 to a wireless network. For those skilled in the art, other steps are possible as well for a device program 102i or operating system 102h to collect data from either (i) a sensor 102f or (ii) a TRE 113 and send the data in a packet without departing from the scope of the present disclosure.

25 Conversely, in order for device 102 to receive a packet or response from server, which could include a challenge or nonce in order to authenticate a device 102 with the server, the physical interface 102a can use a radio to receive the challenge or nonce from a wireless network. The challenge or nonce received from the server through the wireless network could comprise a random number or a pseudo random string of digits, bits, and/or characters. The received data can include
30 information from a server and may also comprise a datagram, a source IP address number, a packet or header value, an instruction for device 102, an acknowledgement to a packet that device 102 sent, a digital signature, and/or encrypted data. The operating system 102h or device driver 102g can use a system bus 102d and SOC 109 to record the received data such as a challenge or nonce from a server in memory such as RAM 102e, and the device program 102i or operating system 102h may
35 access the memory in order to process the received data and determine the next step for the device 102 after receiving the data.

Processing the received data from a server to device 102 could include deciphering or decrypting received data by TRE 113 with a key recorded in TRE 113, sending the challenge or nonce to the TRE 113, reading an instruction from a server, or similar transformations of the received data. The steps within the paragraph above may also describe the steps a device program 102i can perform in order to receive a packet. For those skilled in the art, other steps are possible as well for a device program 102i or device 102 to receive a packet or challenge or nonce from a server without departing from the scope of the present disclosure. A server described herein without the designation of “server 103” or IDS 103 can comprise a different server than server 103 communicating with device 102 in support of an application operating as a device program 102i (e.g. a server supporting mobile payments, TRE 113 operating an embedded SIM, TRE 113 authenticating device 102 with a wireless network, etc.)

Moreover, those skilled in the art will appreciate that the present disclosure may be implemented in other computer system configurations, including hand-held devices, netbooks, portable computers, multiprocessor systems, microprocessor based or programmable consumer electronics, network personal computers, minicomputers, mainframe computers, servers, and the like. The disclosure may also be practiced in distributed computing environments, where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program devices may be located in both local and remote memory storage devices. In addition, the terms “mobile node”, “mobile station”, “mobile device”, “M2M device”, “M2M device”, “networked sensor”, or “industrial controller” can be used to refer to device 102 as contemplated herein.

Although not depicted in Figure 1b, a server shown above in Figure 1a such as IDS server 103 and other servers as well can include equivalent internal components as device 102 in order to operate. The server 103 in Figure 1a could include a processor similar to SOC 109, with primary differences for the processor server being increased speed, increased memory cache, an increased number and size of registers, the use of a 64 bits for datapath widths, integer sizes, and memory address widths, as opposed to an exemplary 32 bits for processor cores in SOC 109 or an exemplary 32 or 16 bits for a processor in TRE 113. Other possibilities exist as well for the size of datapath widths for a TRE 113 and processing core in device 102 without departing from the scope of the present disclosure. Similarly, RAM in a server could be a RAM similar to RAM 102e in device 102, except the RAM in a server could have more memory cells such as supporting exemplary values greater than an exemplary 16 gigabytes, while RAM in device 102 could support fewer memory cells such as less than an exemplary 8 gigabytes. Non-volatile memory for storage in a server 103 in Figure 1a could comprise disks, “solid state drives” (SSDs) or “storage area networks” (SAN) for a server. For a physical interface 102a, in exemplary embodiments, a server 103 in Figure 1a could use a physical, wired LAN interface such as a high-speed Ethernet or fiber optic connection.

In exemplary embodiments, a device 102 can include the functional capabilities of (i) collecting sensor data, (ii) changing state of an actuator 102z, (iii) communicating the data associated with a sensor or actuator with a wireless network, and/or receiving a challenge or nonce from a server and sending a digital signature. The device driver 102g, operating system 102i, and/or device program 102i could optionally be combined into an integrated system for providing the device 102 functionality. Other possibilities exist as well for the configuration or combination of components illustrated in Figure 1b without departing from the scope of the present disclosure.

Figure 1c

Figure 1c is a graphical illustration of a primary platform, in accordance with exemplary embodiments. A primary platform 101 can include a set of PKI keys 101a and a load function 119. Other data and structures can be included in a PP 101 operating within a TRE 113 without departing from the scope of the present disclosure. Note that a TRE 113 operating a PP 101, such as the TRE 113 depicted in Figure 1a and Figure 1b can include electrical components similar to those depicted for a device 101 in Figure 1b, such as including memory, a processor, and an internal data bus. PP 101 can operate on the electrical components within TRE 113, and the exemplary set of PKI keys 101a can be recorded in nonvolatile memory or read only memory of TRE 113. The set of PKI keys 101a for PP 101 can include certificates of (i) CERT.IDS1.ECKA 103c, (ii) CERT.PN.ECDSA 101a-2, (iii) CERT.CI.ECDSA 101a-3, (iv) CERT.CI-OCSF.ECDSA 101a-4, (v) CERT.TSA 101a-5, (vi) CERT.OFL.ECDSA 101a-6, a secret key SK.OFL.ECDSA 101a-7 corresponding to the public key for a certificate of TRE 113 by server 103 or OFL authority 107, and a public key for a TRE manufacturer of PK.TREM.ECDSA 101a-8. Other keys and certificates can be recorded in a set of PKI keys 101a for a PP 101 without departing from the scope of the present disclosure. The set of PKI keys 101a can be use with cryptographic algorithms for both key exchange 101b and load function 119. Note that PP 101 and TRE 113 record the data for PKI keys 101a in a secure manner, such that device 102 cannot read or write data to PKI keys 101a without specific permission (such as device 102 passing TRE 113 certificate 103c from a message 116).

The certificates in a set of PKI keys 101a without leading numeral designations in Figure 1c can also comprise the certificates for a primary platform as specified in the GP OFL document. The use of these PKI keys without leading numeral designations is also described in section 4.1.2.4 of the load function of the GP OFL document. The certificates in a set of PKI keys 101a depicted in Figure 1c with numeral designations of “for 119” can comprise certificates used by a PP 101 that are not specified or contemplated by the GP OFL document. Some certificates in PKI keys 101a could be recorded during manufacturing of TRE 113, such as CERT.PN.ECDSA 101a-2 and CERT.CI.ECDSA 101a-3. Other keys in PP 101 could be subsequently received after distribution of

device 102 with TRE 113 to end users, such as CERT.IDS1.ECKA 103c (which could be received via a message 116).

CERT.CI- OCSP.ECDSA 101a-4 in PKI keys 101a can comprise a certificate for PP 101 used to validate signed time value T2 104e in a step 119a, where an exemplary signed time value T2 104e is depicted in Figure 2a below. CERT.CI- OCSP.ECDSA 101a-4 could be stored or recorded in PKI keys 101a after (i) PP 101 receives data from message 116 via device 102 and (ii) PP verifies CERT.CI- OCSP.ECDSA 101a-4 using CERT.CI.ECDSA 101a-3 (where CI 104 signs CERT.CI- OCSP.ECDSA 101a-4 using SK.CI.ECDSA 104a). Or, in other embodiments CERT.CI.ECDSA 101a-3 could be written along with CERT.CI- OCSP.ECDSA 101a-4 to nonvolatile memory of PP 101 during manufacturing of TRE 113.

Note that CI 104 can use the private key SK.CI-OCSP.ECDSA 104 corresponding to the public key in CERT.CI- OCSP.ECDSA 101a-4 in order to create signed time value T2 104e. In exemplary embodiments, (i) CI 104 uses a different private key than SK.CI.ECDSA 104a to sign time value T2 104e (since a server or computer with SK.CI.ECDSA 104a may be offline and more secured), and consequently (ii) PP 101 uses a public key different from the key in CERT.CI.ECDSA 101a-3 in order to verify signed time value T2 104e such as the signature in Figure 2a, and the different public key can comprise the public key in CERT.CI- OCSP.ECDSA 101a-4. However, PP 101 could use CERT.CI.ECDSA 101a-3 to verify or validate CERT.CI- OCSP.ECDSA 101a-4 (if received in message 116), and thus CERT.CI.ECDSA 101a-3 can be used indirectly to verify signed time value T2 104e.

CERT.TSA 101a-5 in PKI keys 101a can comprise a certificate for PP 101 used to validate signed timestamp T3 108b, where the use of CERT.TSA 101a-5 by PP 101 is depicted and described in connection with Figure 2b below. Timestamp Authority 108 can use a private key of SK.TSA 108a, corresponding to the public key in CERT.TSA 101a-5, in order to create the digital signature 108b for timestamp T3 108b-1. For some exemplary embodiments, the use of a separate certificate CERT.TSA 101a-5 to validate a digital signature 108b for timestamp T3 108b-1 could be omitted, and timestamp T3 108b-1 could be included in data which is signed by a private key corresponding to a different certificate stored by PP 101 in PKI keys 101a. As one example, timestamp T3 108b-1 could be included in data signed by server 103, and PP 101 could rely upon a public key from a verified certificate from server 103 such as a CERT.IDS2.ECDSA 103d in order to validate a digital signature 108b for timestamp T3 108b-1. For embodiments where PP 101 uses a CERT.IDS2.ECDSA 103d in order to validate a digital signature 108b for timestamp T3 108b-1, the received CERT.IDS2.ECDSA 103d can include a signed time value T2 104e-1 and PP 101 can verify the signed time value T2 104e-1 using CERT.CI- OCSP.ECDSA 101a-4 and/or CERT.CI.ECDSA 101a-3.

However, for other embodiments, such as those depicted for a load function 119 in Figure 1c, the use of a separate certificate of CERT.TSA 101a-5 for verifying a digital signature 108b for timestamp T3 108b-1 can be preferred, since relying upon a different certificate (e.g. such as CERT.CI- OCSP.ECDSA 101a-4 and/or CERT.CIECDSA 101a-3) can mean essentially that the timestamp T3 108b-1 is “self signed”. In other words, without (X) a separately signed timestamp T3 108b-1 which is verified by a separate CERT.TSA 101a-5, then (Y) PP 101 may not be able to securely rely upon timestamp T3 108b-1 in order to use signed time value T2 104e-1 to verify that a certificate CERT.IDS1.ECKA 103c or CERT.IDS2.ECDSA 103d has not been revoked.

For example with embodiments alternative to those depicted for a load function 119 in Figure 1c, if timestamp T3 108b is within signed data verified by PP 101 using a certificate such as CERT.IDS2.ECDSA 103d, then PP 101 cannot securely or reliably use a time value T2 104e-1 associated with certificate CERT.IDS2.ECDSA 103d, because that means the signature for timestamp T3 108b-1 is essentially “self signed”. If the certificate for CERT.IDS2.ECDSA 103d has been revoked and the private key associated with CERT.IDS2.ECDSA 103d has been compromised, then (i) the compromised private key associated with CERT.IDS2.ECDSA 103d could be used to sign timestamp T3 108b-1 with an earlier date before revocation of CERT.IDS2.ECDSA 103d, and (ii) an earlier signed timestamp T3 108b-1 from before revocation of CERT.IDS2.ECDSA 103d could be sent in a message 118 from Figure 1a. Consequently, PP 101 could incorrectly determine that CERT.IDS2.ECDSA 103d is valid, when it could have been revoked. Thus, in exemplary embodiments PP 101 can use a recorded CERT.TSA 101a-5 from a set of PKI keys 101a in order to validate a signature 108b for timestamp T3 108b-1.

A primary platform 101 can include a load function 119. The load function 119 depicted in Figure 1c can comprise the load function 119 from Figure 1a with exemplary additional details for conducting signature or certificate verification, ECDH key exchange to derive symmetric ciphering keys, and also decryption with an integrity check. The steps in a load function 119 without leading numeral designations in Figure 1c can also comprise the steps for a primary platform to perform to process a message 118 such as the steps specified in the GP OFL document. The use of these steps without leading numeral designations is also described in section 4.1.2.4 of the load function of the GP OFL document.

The steps depicted in Figure 1c with numeral designations of a leading “119” can comprise steps of the present disclosure that are not specified or contemplated by the GP OLF document in order to securely load firmware 106 while also allowing a PP 101 to determine if a certificate has been revoked. Again, note that there is no description of certificate revocation or the use of the term “revocation” or “revoked” or like descriptions for invalidating a certificate in the GP OLF document. However, support for revocation of a certificate can be essential in order to maintain the security of a system 100 from Figure 1a. Further, conventional technology for determining if a certificate has

been revoked does not support revocation for firmware load with a primary platform, since the primary platform normally does not have the ability to independently keep date and time, especially when power is removed from PP 101. Subsequent descriptions for a load function 119 will focus on the additional steps not contemplated or described by the GP OLF document. Steps for a load function without leading numeral designations can be found within the GP OFL document.

A step 119a for a load function 119 in exemplary embodiments can comprise the depicted step of “VERIFY(CERT 101a-4) [Signed T2 104e]”. PP 101 can use the certificate CERT.CI-OCSP.ECDSA 101a-4 from PP keys 101a in order to verify the signed time value T2 104e-1. The signed time value T2 104e-1 can comprise an interim time value before the expiration time T1 104c where CI 104 confirms that a certificate such as CERT.CI.ECDSA 101a-3 remains valid. In exemplary embodiments, the signed time value T2 104e can comprise a digital signature over a time value T2 104e-1, and the time value T2 104e-1 can comprise the plaintext time value for an end date/time of validity for an OSCP stapling, as depicted and described in connection with Figure 2a.

The signed time value T2 104e-1 can be included as the end of an interim time window for certificate validity such as for the signed time value T2 104e-1 depicted in Figure 2a below. The signed time value T2 104e-1 could be included in an OSCP response 110 value that is “stapled” or included with a certificate such as CERT.IDS1.ECKA 103c or CERT.IDS2.ECDSA 103d which are received by PP 101 via messages 116 and 118, respectively. Consequently, although a message 118 in Figure 1a depicts a message 118 including a digital signature 104e for time value T2 104e-1, the message 118 can also include the time value T2 104e-1. Conducting an exemplary signature verification step is depicted in Figure 2b below for a different signature using a different key, but the same logic and sequence of steps can be used in a step 119a for PP 101 to verify the signed time value T2 104e. Although not depicted in Figure 1c, for embodiment where PP 101 received CERT.CI-OCSP.ECDSA 101a-4 via device 102 from a message 116, PP 101 could also verify CERT.CI-OCSP.ECDSA 101a-4 using CERT.CI.ECDSA 101a-3 before using CERT.CI-OCSP.ECDSA 101a-4 in a step 119a.

As depicted in Figure 1c, upon conclusion of a step 119a, PP 101 can securely read and store the plaintext time value T2 104e-1 (e.g. the time value T2 separate or without a signature). Note that although a single instance of step 119a is depicted for a load function 119 in Figure 1c, a PP 101 could conduct a step 119a for each certificate received, such as, but not limited to CERT.IDS1.ECKA 103c and CERT.IDS2.ECDSA 103d. Consequently, after a step 119c, PP 101 can store and operate with more than one time value T2 104e-1, and subsequently use each of the time values T2 104e-1 with a subsequent step 119c. For the case where the signature verification fails in a step 119a, then PP 101 could abort the load firmware function 119 and report an error condition.

A step 119b for a load function 119 in exemplary embodiments can comprise the depicted step of “Verify(CERT 101a-5) [Signed T3 108b]”. PP 101 can use the certificate CERT.TSA 101a-5 from PP keys 101a in order to verify the digital signature 108b for timestamp T3 108b-1. As contemplated herein, a “time value” and a timestamp can be considered equivalent. The digital signature 108b for a timestamp T3 108b-1 can comprise a timestamp of recent time from a TSA 108 associated with the time for which data in message 118 is processed by server 103. Although not depicted in Figure 1c for a step 119b, a step 119b can comprise PP 101 verifying a digital signature over both the signed time value T3 108b-1 and the transaction ID 117x.

Since PP 101 or TRE 113 may not reliably internally track date and time, the digital signature 108b for a timestamp T3 108b-1 can represent the time PP 101 should use for checking the validity or interim validity of certificates and other time windows. The inclusion of transaction ID 117x within a digital signature 108b for timestamp T3 108b-1 can be useful for PP 101 to verify that the digital signature 108b over at least T3 108b-1 was generated after the PP 101 sends a message 117 with the transaction ID 117x. Upon conclusion of a step 119b, PP 101 can securely read and store the plaintext time value T3 108b-1 (e.g. the timestamp T3 separate or without a signature). As discussed above regarding the use of a CERT.TSA 101a-5, an independent certificate from server 103 can be used to verify the signed timestamp 108b for preferred embodiments. Although not depicted in Figure 1a, the date/time value for the digital signature 108b of timestamp T3 108b-1 can be sent along with a message 118. For other embodiments, a different recorded certificate for PP 101 in PKI keys 101a, such as a PK.IMO.ECDSA (not depicted in Figure 1c, but included in a primary platform in the GP OFL document) could be used to verify digital signature 108b for timestamp T3 108b-1 (assuming the secret key for PK.IMO.ECDSA was used to generate the digital signature 108b for timestamp T3 108b-1. In general, a certificate used to verify a signed timestamp 108b can be from a different entity than server 103, such as from an image owner 105, an OFL authority 107, or a TRE manufacturer.

A step 119c for a load function 119 can comprise PP 101 comparing the plaintext, verified time value T3 108b-1 with each of the time values T2 104e-1. As depicted in Figure 1c, the time value T3 108b-1 should be less than each time value T2 104e-1. Although only a signed time value T2 104e-1 for CERT.IDS2.ECDSA is depicted in Figure 1c, a PP 101 could also use the verified time value T3 108b-1 to verify a different, previous signed time value T2 104e-1 for CERT.IDS1.ECKA 103c from a message 116 in a load function 119. In other words, the timestamp T3 from data in message 118 and associated with received firmware 106 should be before each time value T2 104e-1. For a step 119c, if PP 101 determines that any time value T2 104e-1 is before timestamp T3 108b-1, then PP 101 cannot confirm that a certificate associated with the time value T2 104e-1 remains valid and has not been revoked. If PP 101 determines that any time value T2 104e-1 is before timestamp T3 108b-1, then PP 101 can abort the load 119 function and report the

error condition. Or, PP 101 could respond with an error code and wait for an updated, valid certificate with an updated signed time value T2 104e-1 before completing the load function 119 and begin operating with the firmware 106, such as establishing a firmware session.

After step 119c then PP 101 can conduct a step 119d and proceed with loading firmware 106 and create a firmware session to begin operating with the firmware 106. The conclusion of a step 119d for PP 101 can comprise the exemplary state of loaded firmware in state 3 of loaded firmware in Figure 2-5 of section 2.4 “Loading Procedure” of the GP OLF document. Other examples exist as well for (i) loading firmware 106 after a step 119c to compare time values and (ii) establishing a firmware session in a PP 101 using firmware 106 from a message 118 without departing from the scope of the present disclosure.

Figure 2a

Figure 2a is a simplified response from a certificate authority to verify certificate validity, in accordance with exemplary embodiments. Although a certificate normally has an ultimate expiration time, nodes in a system 100 can need the ability to check that a certificate has not been revoked. As mentioned above, a primary platform recording and using a plurality of different certificates may record and use certificates with long time frames such as several years into the future. For a system 100 potentially hundreds of servers 103 distributed globally and also connected to a public Internet 114, there is a reasonable chance that some certificates for some servers 103 may need to be revoked before the long expiration time of certificates recorded by a PP 101 in a TRE 113. The use of a protocol such as OCSP and an exemplary response 110 depicted in Figure 2a can support a PP 101 determining if a certificate has been revoked.

As depicted and described in connection with Figure 1a, a server 103 could operate a client for requesting a signed, updated status for a certificate. The client could comprise an OCSP client and server 103 could send the request 109 from Figure 1a in order for CI 104 to generate the exemplary response 110 depicted in Figure 2a. CI 104 could operate an OCSP responder 104d from Figure 1a in order to process the request 109 and generate the response 110. The exemplary data depicted for a response 110 is shown using ECDSA algorithms with an SHA-256 hash algorithm, but other algorithms could be utilized as well.

A response 110 in Figure 2a can include the query 109, such that an entity receiving the response 110 can determine the query 109 used to generate the response. As shown, the query could include hash values over the certificate issuer data and a serial number associated with the certificate being checked, such as a serial number for CERT.IDS1.ECKA 103c or CERT.IDS2.ECDSA 103d. The format and logic and algorithms used to create the exemplary data in query 109 within response 110 can be compatible with IETF RFC 6960 and future, related standards, although other, similar

techniques could be used as well to generate an interim confirmation by a certificate issuer that a certificate has not been revoked.

The response data in a response 110 include the certificate issuer information and the certificate serial number, as well as a signed time value T2 104e, which can comprise a digital signature 104e. The plaintext time value T2 104e-1 can be read separately from response 110, although digital signature 104e can be over plaintext time value T2 104e-1. In exemplary embodiments, time value T2 104e-1 can comprise the value of "Next Update" or a time into the future (at the time of CI 104 creating signature 104e) through which a certificate can be considered valid. Further, the status of the certificate can be "good" or "OK" or the equivalent. The response 110 for a certificate such as CERT.IDS1.ECKA 103c or CERT.IDS2.ECDSA 103d can be appended to the CERT.IDS1.ECKA 103c or CERT.IDS2.ECDSA 103d for delivery by server 103 to device 102 and PP 101. In other words, (i) the message 116 with a CERT.IDS1.ECKA 103c can include a first response 110 with a first time value T2 104e-1 and first digital signature 104e and (ii) the message 118 with a CERT.IDS2.ECDSA 103d can include a second response 110 with a second time value T2 104e-1 and a second digital signature 104e. After verifying the signed time value T2 104e, as well as any intermediate certificate associated with the signed time value T2 104e, such as CERT.CI- OSCP.ECDSA 101a-4, PP 101 can use plaintext time value T2 104e-1 to compare with verified timestamp T3 108b to determine if the status in response 110 (e.g. "good") is at the time of timestamp T3 108b. In other words, PP 101 can trust or rely upon the status of a certificate if timestamp T3 108b before the time value T2 104e-1.

Figure 2b

Figure 2b is a flow chart illustrating exemplary steps for verifying a digital signature using PKI keys, parameters, and data input, in accordance with exemplary embodiments. The processes and operations, described below with respect to all of the logic flow diagrams and flow charts may include the manipulation of signals by a processor and the maintenance of these signals within data structures resident in one or more memory storage devices. For the purposes of this discussion, a process can be generally conceived to be a sequence of computer-executed steps leading to a desired result.

These steps usually require physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is convention for those skilled in the art to refer to representations of these signals as bits, bytes, words, information, elements, symbols, characters, numbers, points, data, entries, objects, images, files, or the like. It should be kept in mind, however, that these and similar terms are associated with appropriate

physical quantities for computer operations, and that these terms are merely conventional labels applied to physical quantities that exist within and during operation of the computer.

It should also be understood that manipulations within the computer are often referred to in terms such as listing, creating, adding, calculating, comparing, moving, receiving, determining, configuring, identifying, populating, loading, performing, executing, storing etc. that are often associated with manual operations performed by a human operator. The operations described herein can be machine operations performed in conjunction with various input provided by a human operator or user that interacts with the device, wherein one function of the device can be a computer.

In addition, it should be understood that the programs, processes, methods, etc. described herein are not related or limited to any particular computer or apparatus. Rather, various types of general purpose machines may be used with the following process in accordance with the teachings described herein.

The present disclosure may comprise a computer program or hardware or a combination thereof which embodies the functions described herein and illustrated in the appended flow charts. However, it should be apparent that there could be many different ways of implementing the disclosure in computer programming or hardware design, and the disclosure should not be construed as limited to any one set of computer program instructions.

Further, a skilled programmer would be able to write such a computer program or identify the appropriate hardware circuits to implement the disclosed disclosure without difficulty based on the flow charts and associated description in the application text, for example. Therefore, disclosure of a particular set of program code instructions or detailed hardware devices is not considered necessary for an adequate understanding of how to make and use the disclosure. The inventive functionality of the claimed computer implemented processes will be explained in more detail in the following description in conjunction with the remaining Figures illustrating other process flows.

Further, certain steps in the processes or process flow described in all of the logic flow diagrams below must naturally precede others for the present disclosure to function as described. However, the present disclosure is not limited to the order of the steps described if such order or sequence does not alter the functionality of the present disclosure. That is, it is recognized that some steps may be performed before, after, or in parallel other steps without departing from the scope and spirit of the present disclosure.

The processes, operations, and steps performed by the hardware and software described in this document usually include the manipulation of signals by a CPU or remote server and the maintenance of these signals within data structures resident in one or more of the local or remote memory storage devices. Such data structures impose a physical organization upon the collection of data stored within a memory storage device and represent specific electrical or magnetic elements. These symbolic representations are the means used by those skilled in the art of computer

programming and computer construction to most effectively convey teachings and discoveries to others skilled in the art.

A step 119a, and related steps for verifying a digital signature in the present disclosure, can use an ECC based elliptic curve digital signature algorithm (ECDSA) algorithm, and other possibilities exist as well without departing from the scope of the present disclosure. PP 101 can use a signature verification in step 119a in order to validate a signature for a received timestamp 108b. When using ECDSA algorithm in non-deterministic mode for a signature creation, a value of “k” or “r”, which could comprise a random number that can be included with the digital signature 108b. When using a DSA or ECDSA in deterministic mode, such as specified in IETF RFC 6979 and titled “Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)”, which are hereby incorporated by reference, then the requirement for a separately transmitted random number with a digital signature such as signed timestamp 108b (such including as value “k” or “r”) can be optionally omitted, such that “r” can be deterministically calculated based on the message to sign.

In exemplary embodiments, server 103 and other servers in Figure 1a can utilize deterministic ECDSA without also sending a random number along with a digital signature, although the value “r” from the deterministic mode could be sent with the digital signature such as signed timestamp 108b. In other words, a value can be sent with the digital signature such as signed timestamp T3 108b that is deterministically derived and associated with the message to sign. In other exemplary embodiments, a random number can be generated as a derived value for the random number such as “r” sent with a digital signature such as signed timestamp T3 108b.

For a signature verification step 119b, the exemplary message to verify comprises the plaintext timestamp value T3 108b-1, which can be decrypted by PP 101 using a load function 119. Although a load function 119 in Figure 1c above depicts the timestamp value T3 108b-1 as decrypted from a message M4, for some embodiments the timestamp value T3 108b-1 could be sent as plaintext in a message 118. In other words, the timestamp value T3 108b-1 does not need to be encrypted using an encrypted message M4 since a digital signature 108b for the timestamp value T3 108b-1 can separately be included with a message 118 as depicted in Figure 1a.

The message to verify values can be transmitted to PP 101, such as shown for timestamp T3 108b-1 and the digital signature 108b for timestamp T3. Note that in some exemplary embodiments for a step 119b, the message to verify can include both the timestamp T3 108b-1 and the transaction ID 117x, where PP 101 generates the transaction ID 117x for creation of the message 117 as depicted and described in connection with Figure 1a. The message to verify values can be input into a message digest algorithm 230, which could comprise a standard algorithm such as SHA-256, SHA-3, or similar algorithms. The output of message digest algorithm 230 can be input along with parameters and a public key from a certificate such as CERT.TSA 101a-5 into signature verification

algorithm 228. Parameters can specify encoding rules, padding, key lengths, selected algorithms, curve names, and other values or fields necessary to utilize a signature algorithm 228. Parameters can be specified in the certificate. Both a signature creation step and a signature verification step use the same or equivalent values for parameters. Private key SK.TSA 108a can comprise the private
5 key used to create the digital signature 108b for at least the timestamp T3 108b-1. Although not depicted in Figure 2b, a random number for values such as “k” and “r” for ECDSA and related signatures could be input as well into a signature algorithm 228.

A signature algorithm 228 for a step 119b can also use input of the received digital signature 108b for timestamp 108b-1, which can be received from a message 118. The calculation of a
10 digital signature 108b using the public key from a certificate such as CERT.TSA 105a-1 can be compared with the received, digital signature 108b. If the two values for the calculated version of the digital signature 108b and the received version of the digital signature 108b (from message 118) match or are equal, then the received, signed timestamp T3 108b-1 can be considered valid and the plaintext timestamp value of T3 108b-1 can be used with a comparison step 119c and T2 104e-1. If
15 the two values for the calculated version of the digital signature 108b and the received version of the digital signature 108b (from message 118) do not match or are not equal, then the received, signed timestamp T3 108b-1 can be considered invalid and subsequently not used. PP 101 can respond with an error code or status to request a new, different signed timestamp T3 108b-1 and stop the firmware load function 119.

Conclusion

Various exemplary embodiments have been described above. Those skilled in the art will understand, however, that changes and modifications may be made to those examples without departing from the scope of the claims.

CLAIMS

1. A method for a tamper resistant element (TRE) to securely receive firmware, the method comprising the TRE:

5 recording a certificate issuer (CI) certificate, a TRE certificate, and a timestamp authority certificate;

receiving from an open firmware loader (OFL) agent, a first server certificate for a server, wherein the first server certificate includes a signed first time value, and wherein the TRE uses at least the CI certificate to verify the signed first time value;

10 deriving an ephemeral PKI key pair comprising an ephemeral public key and an ephemeral private key;

conducting an elliptic curve Diffie Hellman key exchange with the ephemeral private key and a server public key from the first server certificate in order to derive a symmetric ciphering key;

15 encrypting the TRE certificate using at least the symmetric ciphering key;

sending to the OFL agent the encrypted TRE certificate;

receiving from the OFL agent a message with the firmware, a signed timestamp, and a second server certificate for the server, wherein the second server certificate includes a signed second time value, and wherein the TRE uses at least the CI certificate to verify the signed
20 second time value;

verifying the signed timestamp using at least the timestamp authority certificate;

comparing (i) the signed first time value with the signed timestamp in order to confirm validity of the first server certificate and (ii) the signed second time value with the signed timestamp in order to confirm validity of the second server certificate; and

25 loading the firmware for the TRE.

2. The method of claim 1, wherein the first server certificate includes a first online certificate status protocol (OCSP) signature for the signed first time value, and wherein the second server certificate includes a second OCSP signature for the signed second time value.

3. The method of claim 2, wherein the signed first time value comprises a first next update value for an OCSP protocol, and wherein the signed second time value comprises a second next update value for the OCSP protocol.

4. The method of any one of claims 1 to 3, further comprising receiving from the OFL agent the message with a certificate for an OCSP server, wherein the TRE uses the certificate for the OCSP server to verify the signed second time value, and wherein the TRE uses the CI certificate to verify the certificate for the OCSP server.

5. The method of any one of claims 1 to 4, wherein a system on a chip includes the TRE, and wherein the TRE operates a primary platform for loading the firmware.

6. The method of any one of claims 1 to 5, further comprising sending to the OFL agent the encrypted TRE certificate and a transaction identity, and verifying the signed timestamp using at least the timestamp authority certificate and the transaction identity.

7. The method of any one of claims 1 to 6, further comprising, sending, by the OFL agent, the encrypted TRE certificate to an image delivery server (IDS), and receiving, by the OFL agent, the message from the IDS.

8. The method of claim 7, further comprising, receiving, by the IDS (i) the signed timestamp from a timestamp authority, and (ii) the firmware from an image owner.

9. A method for a tamper resistant element (TRE) to securely receive firmware, the method comprising the TRE:

recording a certificate issuer (CI) certificate, a TRE certificate, and a timestamp authority certificate;

receiving from an open firmware loader (OFL) agent, a first server certificate for a server, wherein the first server certificate includes a first online certificate status protocol (OCSP) signature for a first time value, and wherein the TRE uses at least the CI certificate to verify the first OCSP signature;

5 generating a transaction identity (ID);

 sending to the OFL agent at least the TRE certificate and the transaction ID;

 receiving from the OFL agent a message with the firmware, a digital signature, and a second server certificate for the server, wherein the second server certificate includes a second OCSP signature for a second time value, wherein the TRE uses at least the CI certificate to
10 verify the second OCSP signature, and wherein the digital signature is generated over at least the transaction ID and a third time value;

 verifying the digital signature using at least the timestamp authority certificate and the transaction ID;

 comparing (i) the first time value with the third time value order to confirm validity of
15 the first server certificate and (ii) the second time value with the third time value in order to confirm validity of the second server certificate; and

 loading the firmware for the TRE.

10. The method of claim 9, further comprising receiving from the OFL agent the message with the third time value.

20 11. The method of claim 9 or 10, wherein the first time value comprises a first next update value for an OCSP protocol, and wherein the second time value comprises a second next update value for the OCSP protocol.

 12. The method of any one of claims 9 to 11, further comprising receiving from the OFL agent the message with a certificate for an OCSP server, wherein the TRE uses the
25 certificate for the OCSP server to verify the second OCSP signature, and wherein the TRE uses the CI certificate to verify the certificate for the OCSP server.

13. The method of any one of claims 9 to 12, wherein a system on a chip includes the TRE, and wherein the TRE operates a primary platform for loading the firmware.

14. The method of any one of claims 9 to 13, further comprising, sending, by the OFL agent, the TRE certificate to an image delivery server (IDS), and receiving, by the OFL agent, the message from the IDS.

15. The method of claim 14, further comprising, receiving, by the IDS (i) the digital signature from a timestamp authority, and (ii) the firmware from an image owner.

16. A method for a server to securely transmit firmware to a device, the method comprising the server:

recording (i) a first server private key for a first server certificate and (ii) a second server certificate;

using an online certificate status protocol (OCSP) client to receive a first signed timestamp for the first server certificate and a second signed timestamp for the second server certificate;

sending, to the device, the first server certificate and at least the first signed timestamp; receiving a message from the device, wherein the message comprises an ephemeral public key and a first ciphertext with a transaction identity (ID);

conducting a first elliptic curve Diffie Hellman key exchange with the ephemeral public key and the first server private key in order to derive a first symmetric ciphering key;

decrypting the first ciphertext using at least the first symmetric ciphering key;

sending the transaction ID to a timestamp authority;

receiving, from the timestamp authority, a third signed timestamp and a first digital signature, wherein the first digital signature is over at least the transaction ID and the third signed timestamp, and wherein the third signed timestamp is less than both the first signed timestamp and the second signed timestamp;

deriving a second server private key and a second server public key;

conducting a second elliptic curve Diffie Hellman key exchange with the ephemeral public key and the second server private key in order to derive a second symmetric ciphering key;

encrypting the firmware with the second symmetric ciphering key;

5 generating a second digital signature for the second server public key, wherein the second digital signature is verified using the second server certificate; and

sending, to the device, the encrypted firmware, the second server public key, the second server certificate, the second signed timestamp, the third signed timestamp, the first digital signature, and the second digital signature.

10 17. The method of claim 16, wherein the first signed timestamp comprises a first next update value for an OCSP protocol, and wherein the second signed timestamp comprises a second next update value for the OCSP protocol.

15 18. The method of claim 16 or 17, further comprising, sending, to the device, the first server certificate with a certificate for an OCSP server, wherein the certificate for the OCSP server is verified using a certificate issuer (CI) certificate.

19. The method of any one of claims 16 to 18, wherein the first server certificate and the second server certificate are verified using a certificate issuer (CI) certificate.

20. The method of any one of claims 16 to 19, wherein the server comprises an image delivery server (IDS), and wherein the IDS receives the firmware from an image owner.

Figure 1a

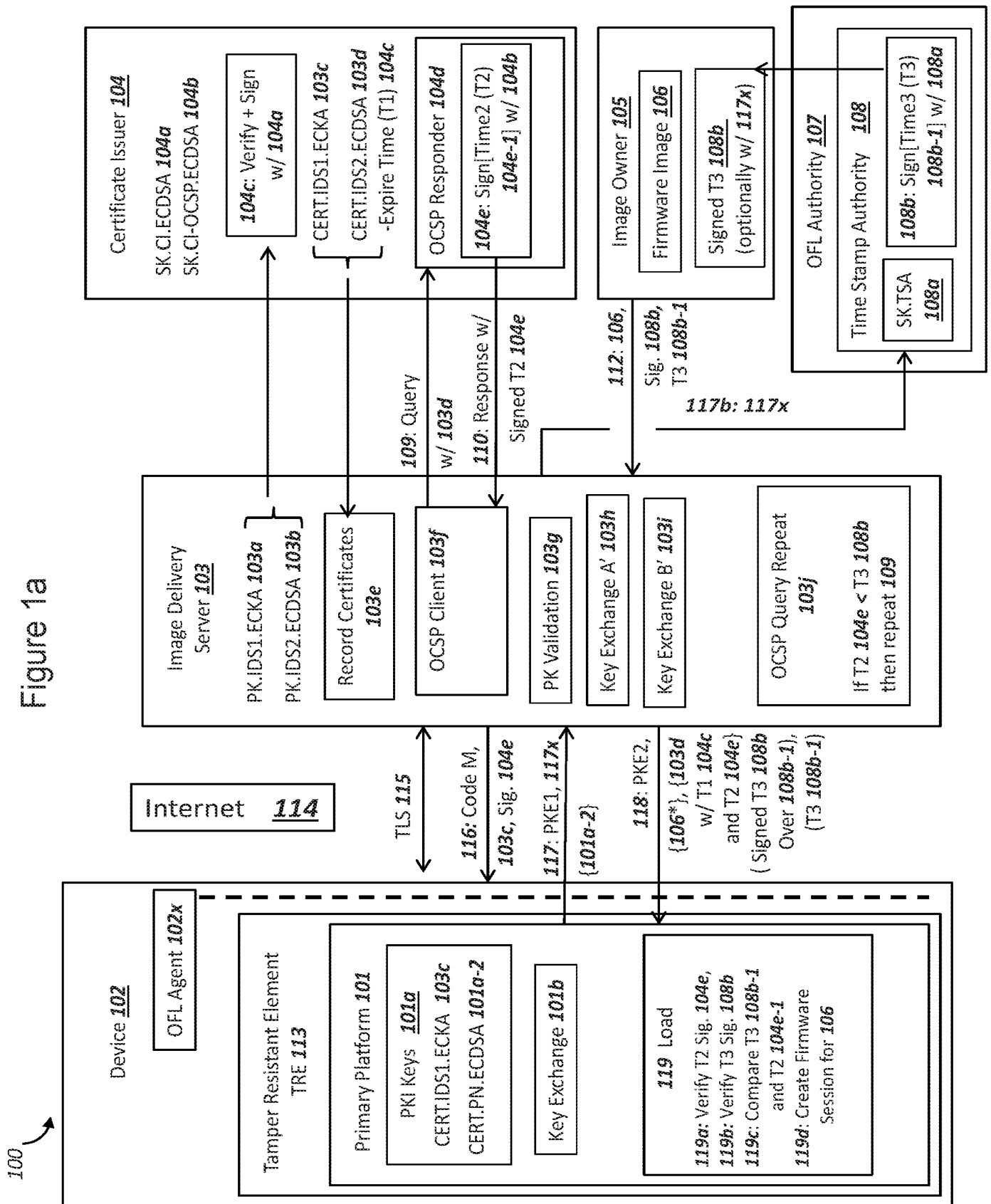


Figure 1c

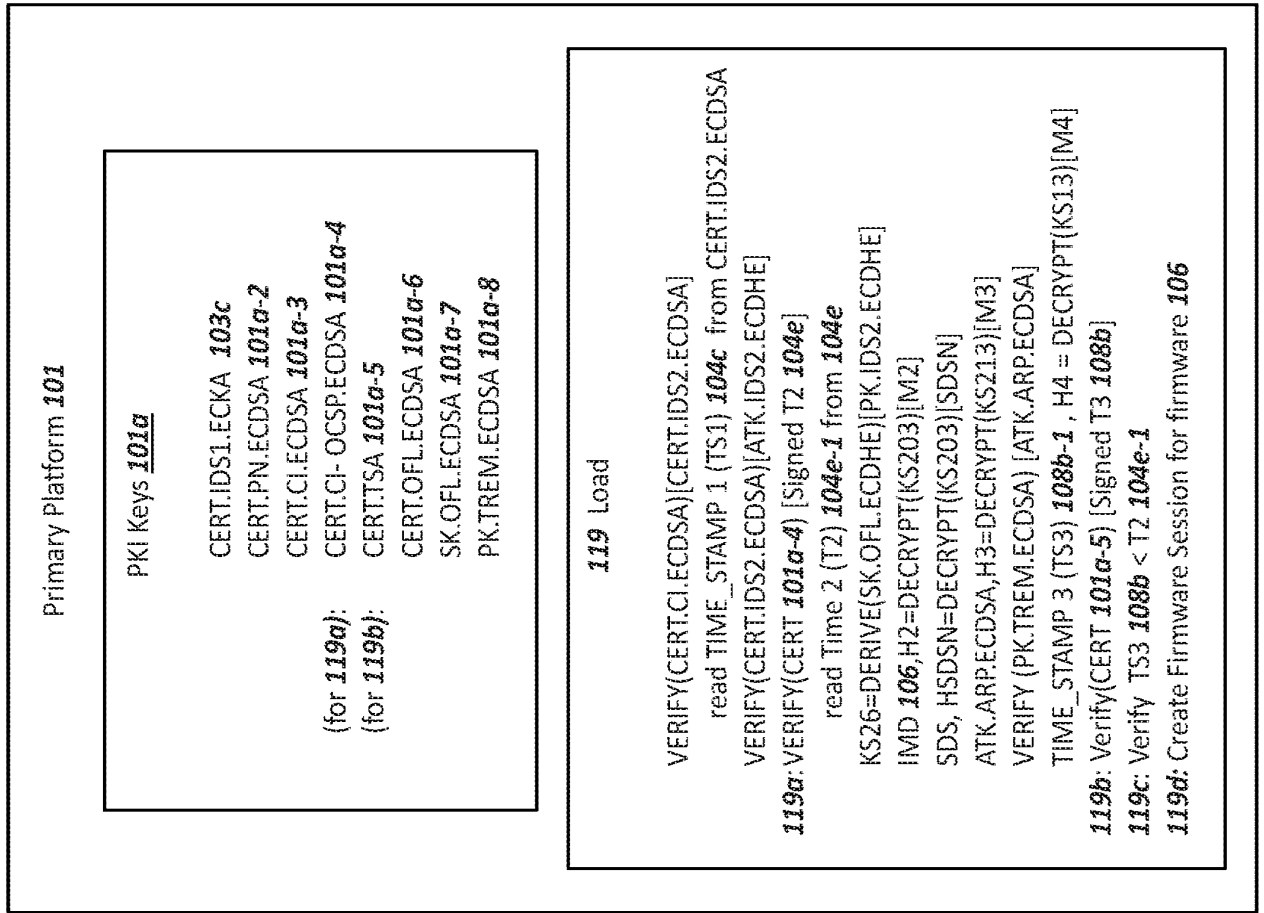


Figure 1b

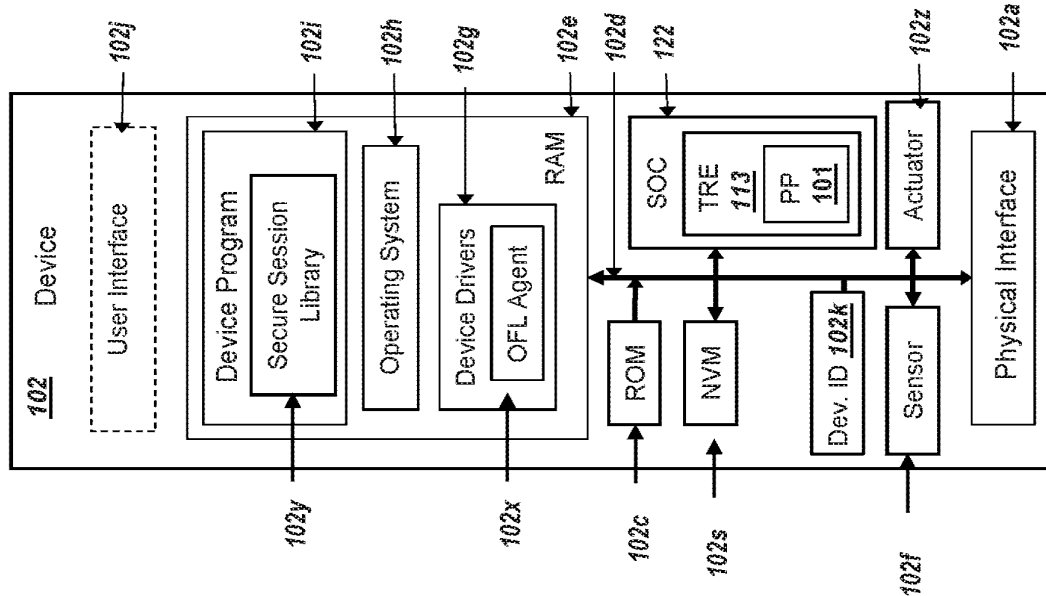


Figure 2a**OCSP Response 110**

Verifying CERT.IDS2.ECDSA with SHA 256 Algorithm ("... " is long text removed)

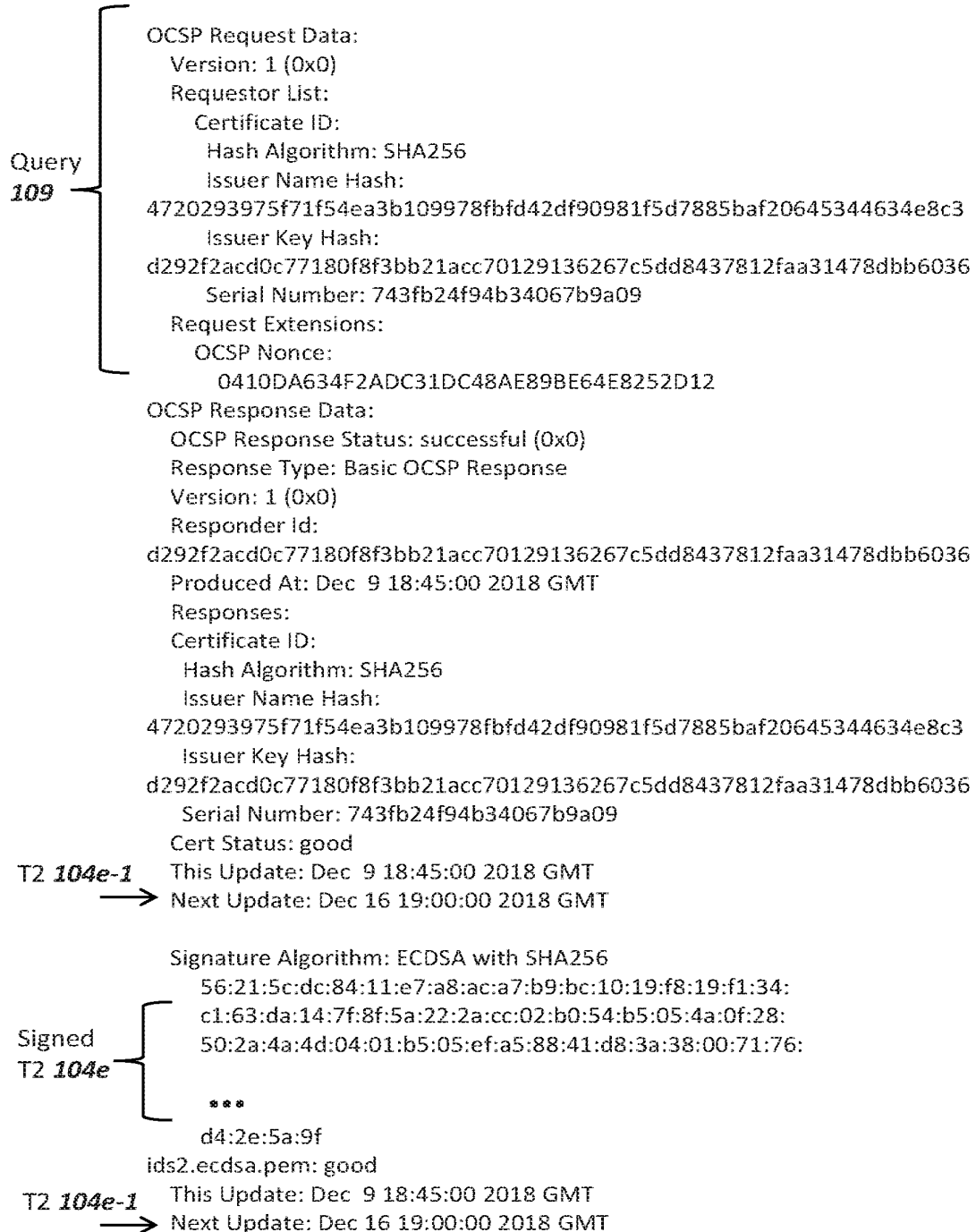
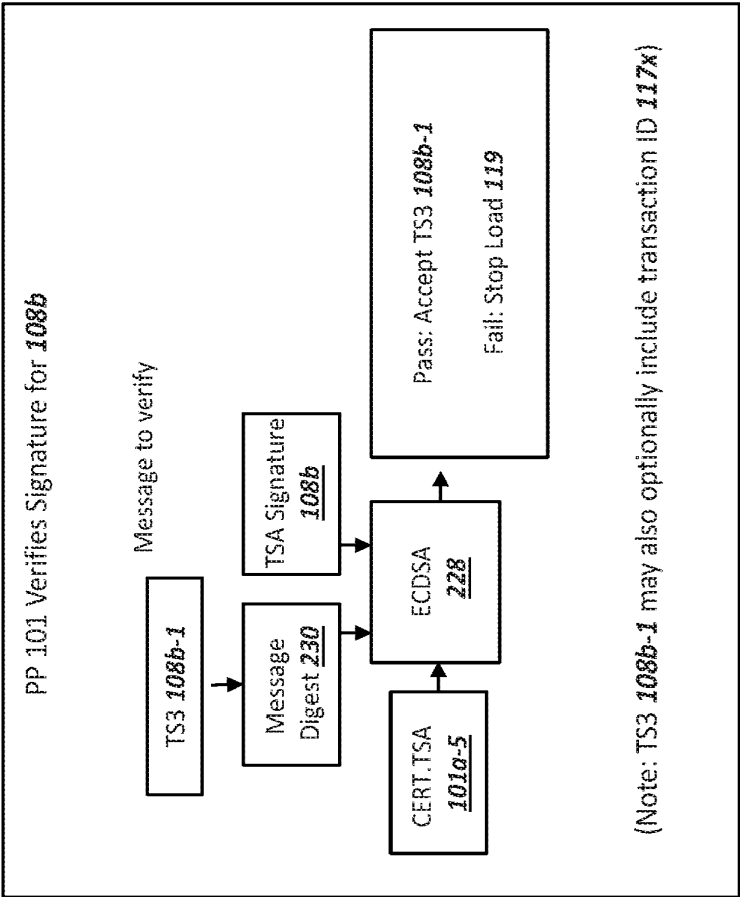


Figure 2b

119b – Time Stamp 3 (TS3) 108b Verification



INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 2019/066257

A. CLASSIFICATION OF SUBJECT MATTER

*G06F 21/64 (2013.01)**H04L 9/14 (2006.01)*

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

PatSearch (RUPTO internal), USPTO, PAJ, K-PION, Esp@cenet, Information Retrieval System of FIPS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6009177 A (CERTO LLC) 28.12.1999, col.1, lines 58-60, col.2, lines 4-7, 46-55, col.3, lines 23-36, col.4, lines 59-62, col.11, lines 18-42, col.13, lines 1-6, col.14, lines 29-45, col.21, lines 47-54, col.23, lines 13-30, col.24, lines 1-3, col.27, lines 29-32, col.37, lines 30-35, 60-64, col.38, lines 22-36, col.39, lines 2-6, 24-32, col.42, lines 11-13, col.44, lines 41-44, col.48, lines 51-62	1-4, 9-11, 16-18
A	US 2016/0065370 A1 (ERIC LE SAINT et al) 03.03.2016	1-4, 9-11, 16-18
A	US 7177839 B1 (CERTCO INC) 13.02.2007	1-4, 9-11, 16-18
A	US 2006/0080536 A1 (TIME CERTAIN LLC) 13.04.2006	1-4, 9-11, 16-18



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“E” earlier document but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search

24 March 2020 (24.03.2020)

Date of mailing of the international search report

09 April 2020 (09.04.2020)

Name and mailing address of the ISA/RU:

Federal Institute of Industrial Property,
Berezhkovskaya nab., 30-1, Moscow, G-59,
GSP-3, Russia, 125993

Facsimile No: (8-495) 531-63-18, (8-499) 243-33-37

Authorized officer

A. Kulikov

Telephone No. (495) 531-65-15

Form PCT/ISA/210 (second sheet) (January 2015)

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 2019/066257

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:
2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
3. ☒ Claims Nos.: 5-8, 12-15, 19, 20
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- ☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- ☐ No protest accompanied the payment of additional search fees.

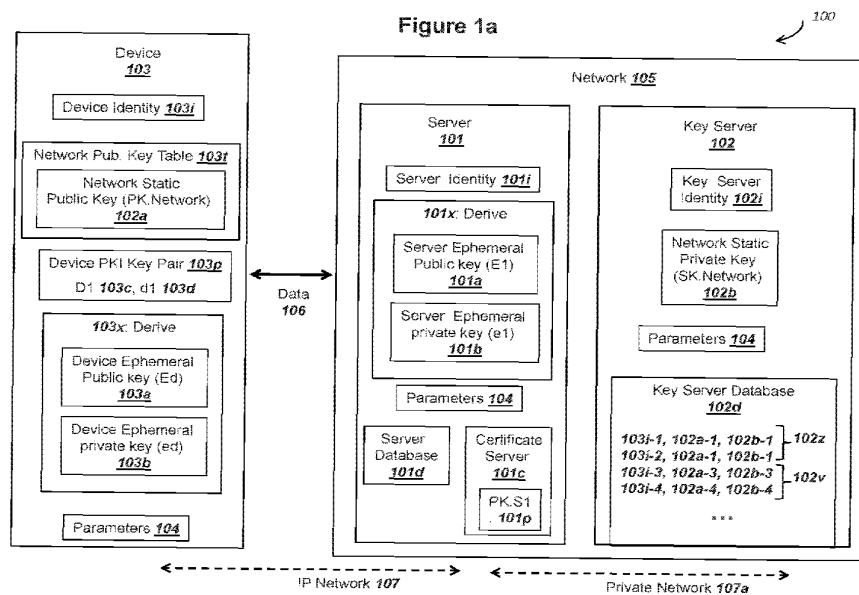
(51) International Patent Classification:
H04L 9/00 (2006.01)(21) International Application Number:
PCT/US2019/037911(22) International Filing Date:
19 June 2019 (19.06.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/687,411 20 June 2018 (20.06.2018) US(71) Applicant: **IOT AND M2M TECHNOLOGIES, LLC**
[US/US]; 807 Davis Street, Unit 2207, Evanston, IL
60201-7104 (US).(72) Inventor: **NIX, John, A.**; c/o IoT and M2M Technologies, LLC, 807 Davis Street, Unit 2207, Evanston, IL
60201-7104 (US).(74) Agent: **THOMSON, Kirsten, L.**; McDonnell Boehnen
Hulbert & Berghoff LLP, 300 South Wacker Drive, Chicago,
IL 60606 (US).(81) Designated States (*unless otherwise indicated, for every
kind of national protection available*): AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ,
CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO,
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,
HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP,
KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME,
MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ,
OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA,
SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN,
TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(54) Title: AN ECDHE KEY EXCHANGE FOR SERVER AUTHENTICATION AND A KEY SERVER



(57) **Abstract:** A server can receive a device public key and forward the device public key to a key server. The key server can perform a first elliptic curve Diffie-Hellman (ECDH) key exchange using the device public key and a network private key to derive a secret X1. The key server can send the secret X1 to the server. The server can derive an ECC PKI key pair and send to the device the server public key. The server can conduct a second ECDH key exchange using the derived server secret key and the device public key to derive a secret X2. The server can perform an ECC point addition using the secret X1 and secret X2 to derive a secret X3. The device can derive the secret X3 using (i) the server public key, a network public key, and the device private key and (ii) a third ECDH key exchange.

[Continued on next page]

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

AN ECDHE KEY EXCHANGE FOR SERVER AUTHENTICATION AND A KEY SERVER

CROSS-REFERENCE TO RELATED APPLICATION

This international PCT application claims the benefit of the filing date of U.S. Provisional
5 Patent Application Serial No. 62/687,411, filed June 20, 2018, which is hereby incorporated by
reference in its entirety.

BACKGROUND**Technical Field**

The present systems and methods relate to conducting an ephemeral elliptic curve Diffie
10 Hellman key exchange (ECDHE) with authentication and multiple parties, and more particularly to
communications between a computing device, a server, and a key server over a network in order for
the computing device and the server to mutually derive a symmetric ciphering key with authentication
of the server.

15 Description of Related Art

The use of elliptic curve cryptography (ECC) for computing devices has expanded over the
past decade and is also expected to continue to grow. Many applications use or propose using
ephemeral elliptic curve Diffie Hellman (ECDHE) key exchanges in order to derive a symmetric
ciphering key. Prominent examples today include embedded universal integrated circuit cards
20 (eUICCs) also known as embedded SIMs, Transport Layer Security (TLS) version 1.3 from the
Internet Engineering Task Force (IETF), and the Device Provisioning Protocol (DPP) from the WiFi
Alliance™. Other examples are expected in the future as well, such as the use of ECDHE in order to
protect the Subscription Permanent Identifier (SUPI) for 5G mobile networks, where the SUPI is
equivalent to an International Mobile Subscriber Identity (IMSI). ECDHE can be considered a subset
25 of elliptic curve Diffie-Hellman key exchanges (ECDH), where ECDHE key exchanges use at least
one ephemeral on short-term elliptic curve PKI key pair. Applications use ECDHE key exchanges in
order for two nodes to mutually derive a symmetric ciphering key and a message authentication code
(MAC) key using a key derivation function. The symmetric ciphering key can subsequently be used
with a symmetric ciphering algorithm such as the Advanced Encryption Standard (AES) and the MAC
30 key can be used to verify message integrity. In this manner, secure communication can be established
between two nodes.

ECDHE key exchanges depend on a first node deriving a first ephemeral private and public key pair and a second node deriving or using a second private and public key, where the public key infrastructure (PKI) keys use a common elliptic curve. The elliptic curve can be specified in parameters that define a named curve such as secp256r1 (p256), secp256k1, secp385r1, etc., and many other possibilities exist as well. ECDHE key exchanges have multiple benefits over older generation technology such as Diffie Hellman key exchanges. With ECDHE, elliptic curve cryptography can be utilized with shorter keys and faster processing times compared to previous technology, for the equivalent level of security or bit length of keys. For example, a 256 bit ECC PKI key pair can be used to obtain a comparable level of security as that obtained from using a 3072 bit RSA based PKI key pair. Calculation or processing time for conducting an ECDHE key exchange can also be faster than a traditional Diffie Hellman key exchange for the same level of security, as defined by the resulting key length of a derived shared secret from the key exchange.

Although the use of ECDHE key exchanges is growing rapidly, improvements can be made for ECDHE key exchanges in order to further enhance security and also leverage existing keys that may be recorded by the nodes participating in an ECDHE key exchange. As one example, an ECDHE key exchange as contemplated for (a) the exemplary applications and standards from two paragraphs above do not normally (b) provide authentication of either node. Separate steps than an ECDHE key exchange have to be conducted in order to authenticate endpoints, such as using an elliptic curve digital signature algorithm (ECDSA) with static or long-term ECC PKI keys recorded by the nodes. ECDSA algorithms also have challenges, where the reuse of a value k for two different signatures can reveal the private key. As another example and related to the authentication issue above, an ECDHE is susceptible to "man in the middle" attacks, where an intermediate node or different node than the intended node can perform the ECDHE key exchange instead of the intended node. Thus, although ECDHE can securely establish a symmetric ciphering key for confidentiality of data communications, the confidentiality could be established with a party or node that is not the intended recipient of the confidential communications. Consequently, a need exists in the art for the intended two nodes for confidential communications to use an ECDHE key exchange in a manner where at least one of the two nodes can be authenticated.

A primary goal of ECDHE key exchanges is also to obtain forward secrecy, where an ECDHE key exchange can periodically be re-conducted in order to rotate or re-establish a new symmetric ciphering key. In this manner, if a private key is compromised then only the subset of historical data encrypted using the compromised private key is subject to decryption, and other communications using a different private key can remain secured. An authenticated ECDH key exchange can be conducted using at least one static PKI key pair (e.g. not an ephemeral key exchange with ephemeral PKI keys), but without the benefits of forward secrecy. A need exists in the art where two parties can

conduct an authenticated ECDHE key exchange (e.g. by using ephemeral PKI keys) in order to obtain the benefits of forward secrecy.

The use of ECDH key exchanges (e.g. with at least one static PKI key pair) is also subject to greater security risks over time, where repeated use of one static PKI key pair is subject to cryptographic analysis and “leakage” of equivalent bits of security over time. Further, the use of ECDH key exchanges with one static PKI key pair and one ephemeral PKI key pair is more subject to risks of attacks from specifically chosen ephemeral PKI keys, such as ephemeral public keys that are either (i) not on the curve or (ii) specifically selected to expose information about the static private key. Thus, (a) repeated use of ECDHE key exchanges over time with different ephemeral PKI keys, compared to (b) using an ECDH key exchange with one static PKI key pair will result in greater security regarding confidentiality of communications. A need exists in the art where the greater security of ECDHE key exchanges can be obtained while also using static ECC PKI keys recorded by at least one of the nodes deriving a symmetric ciphering key using the ECDHE key exchange.

Many applications or new standards such as TLS version 1.3, DPP version 1.0 and 5G network standards from the 3rd Generation Partnership Project (3GPP) implement ECDHE key exchanges in order to quickly establish confidentiality early in the communications between two nodes. As noted above, a traditional ECDHE key exchange establishes confidentiality without authentication, and authentication must be obtained through other means, such as ECDSA or DSA with certificate verification, message digest, etc. However, the nodes participating in communications with the above standards typically have access to other, secure and previously recorded PKI keys besides the ephemeral PKI keys derived in order to conduct the ECDHE key exchange. A need exists in the art for a node to use the previously recorded PKI keys for (a) a new ECDHE key exchange in order to establish an authenticated key exchange without (b) the risks of ECDH key exchanges for static PKI keys as discussed above.

In addition, a device seeking to establish secured communication with a server may not be able to efficiently or securely verify a full certificate chain for a certificate from a server or a network, due to limitations such as lack of Internet connectivity for the device, lack of global date and time to properly check certificate revocation, incompatible parameters for verifying signatures from intermediate certificate authorities, etc. A need exists in the art for a device to use a previously recorded public key for a server or a network in order to include the public key in an authenticated ECDHE key exchange such that communications with a server can be secured without a separate requirement for full certificate verification through all intermediate root certificates to a root certificate stored in the device.

Solutions have been proposed in the art for an authenticated Diffie-Hellman or elliptic curve Diffie-Hellman key exchange using ephemeral keys and static keys. Blake-Wilson et al in the paper “Key Agreement Protocols and their Security Analysis”, which is herein incorporated by reference,

propose the use of both long-term static keys and short-term ephemeral keys with a DH key exchange in order to conduct the key exchange in an authenticated manner in order to address some needs in the art mentioned above. Likewise, the Internet Engineering Task Force (IETF) proposes the use of elliptic curve ephemeral and static PKI keys in the “Request for Comments” (RFC) 5753 document
5 “Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)”, which is also hereby incorporated by reference.

However, the methods described for Blake-Wilson, RFC 5753, and related systems depend on (a) the recipient/responder of an ephemeral ECC public key from a sender/initiator to (b) also to record or operate with the static private ECC key corresponding to the static public key recorded by the
10 sender. This can reduce scalability of a system with (i) a plurality of senders/initiators and (ii) a plurality of recipients/responders receiving an ephemeral ECC public keys for ECDHE key exchanges, since each recipient/responder also needs to record and operate on the static ECC private key corresponding to the static ECC public key recorded by the sender/initiator. The overall security of a system can be decreased for a system of potentially millions of devices and several servers, where the
15 servers need to record server static private ECC keys corresponding to server static public ECC keys recorded by devices. A need exists in the art for (a) a recipient/responder to support authenticated ephemeral ECDH key exchanges without (b) the recipient/responder also recording the static ECC private key corresponding to the static ECC public key recorded by the sender/initiator.

Many other examples exist as well for needs in the art to conduct an ECDHE key exchange in
20 a secure manner where at least one of the nodes can be authenticated, and the above are examples are just a few and intended to be illustrative instead of limiting.

SUMMARY

Methods and systems are provided for a server to conduct an ephemeral elliptic curve Diffie-Hellman key exchange (ECDHE) with a device and a key server. The device and the server can record and operate a set of compatible values and algorithms for a key pair generation algorithm, an ECDH key exchange algorithm, a key derivation function, a symmetric ciphering algorithm, and a random number generator, and a set of cryptographic parameters. The device can comprise a computing device with a network interface to communicate with the server via an IP network. The device can comprise a transducer device for operating a transducer and communicating the transducer data with the server via secured communications. The device can comprise a device for “the Internet of Things”, a mobile phone, a tracking device, a security system, a module, or similar devices. The server can comprise a computing device with a network interface to communicate with the device via the IP network and the key server via a private network. The device can record a network static public key and a domain name service (DNS) name or uniform resource locator (URL) for the server. The key server can record the network static private key. The server can record and operate a server database. The device can be one of a plurality of different devices communicating with the server.

Before distribution to an end user of the computing device, a device manufacturer or a device distributor or a device owner could record a set of data in nonvolatile memory for the device. In addition to regular operating data and programs for the device, such as an operating system and a transducer driver, the data recorded in device before distribution could include (i) a network static public key, (ii) a set of cryptographic parameters associated with the network static public key, and (iii) a device identity for the computing device. For a first exemplary embodiment, the network static public key can be unique for the device and not shared with other devices. For a second exemplary embodiment, the network static public key can be shared across a set of devices and thus the network static public key would not be uniquely recorded in an individual device, but the network static public key could be uniquely recorded in a set of devices.

After power up and/or connecting with the IP network, the device can use the random number generator, the cryptographic parameters, and the key pair generation algorithm to derive a device ephemeral private key and a device ephemeral public key. The device can send the device ephemeral public key and the associated set of cryptographic parameters to the server in a first message using the DNS name or a URL for the server. The device can also optionally send a device identity or a secure hash value for the network static public key to the server, in order for the server to identify the device or set of devices. In some exemplary embodiments, the identity of the device and also the secure hash value can be omitted from the first message and the server identifies the group of devices by a particular IP address and port number and/or URL and/or DNS name used by the server and receiving

data from devices. In other words, without identifying data, (X) a subset of devices sending data (i) to the server where (ii) the server uses a particular name, URL, or IP address and/or port number can be identified by (Y) the server receiving data from the devices using the IP address and port number and/or URL and/or DNS name.

5 The server can receive the first message and process the first. The server can use the received set of cryptographic parameters to conduct an ECC point validation step to verify that the received ECC public key comprises a point on a named curve specified by the set of cryptographic parameters. The server can record the name or URL for a key server and communicate with the key server through a private network or a secured session over a public network such as the Internet. The server can
10 establish a secure session with the key server. The server can (a) select the key server for the device using identifying information from the first message and then (b) forward the device ephemeral public key in a second message to the key server after the validation step. The second message can also include the set of cryptographic parameters. The identifying information from the first message for
15 the device could comprise any of (i) an optional identity of the device in the first message, (ii) an optional secure hash value for the network static public key in the first message, (iii) the use of a particular set of cryptographic parameters in the first message, where the set of cryptographic parameters are associated with a particular key server, or (iv) the server can operate such that the use
of a particular URL or IP address and port number is mapped to a particular key server.

 The server can use a random number generator and a key pair generation algorithm and the set
20 of cryptographic parameters to derive a random number for a server ephemeral private key and then use the server ephemeral private key to generate a server ephemeral public key. The server can conduct a first elliptic curve Diffie-Hellman key exchange (ECDH) using (i) the derived server ephemeral private key and received device ephemeral public key and (ii) the set of cryptographic parameters in order to derive a first shared secret. The server can also operate and record a key
25 derivation function and a symmetric ciphering algorithm. The server can operate or be associated with a server database in order to record data for the server communicating with a plurality of different devices, such that different keys for different devices could be tracked by the server. In exemplary embodiments the first message is received with a random number generated by the device and also a source IP address and port number, and the server records the random number and the source IP
30 address and port number for the first message in the server database.

 The key server can receive the second message from the server over the secure connection. The second message can include the device ephemeral public key and the set of cryptographic parameters. For embodiments where the first message includes identifying information for the device (e.g. any of (i) through (iv) in the above paragraph), then the second message to the key server can
35 also include the identifying information for the device. The key server can select or read the network static private key using the second message received from the server (including possibly identifying

information of the device such as, but not limited to, a secure hash value for the network static public key, to select a specific network static private key for the device). The key server can conduct a second ECDH key exchange using (i) the selected network static private key for the device and the received device ephemeral public key and (ii) the set of cryptographic parameters in order to derive a second shared secret. The key server can send a response to the second message in the form of a third message to the server, where the third message includes the derived second shared secret.

The server can receive the derived second shared secret from the key server in the third message. The third message can also include identifying information such that the server can track which of the devices the third message from the key server is associated with. The server can conduct an ECC point verification step to verify that the received point from the key server comprising the second shared secret is a point on the ECC curve specified by the set of cryptographic parameters received in the first message. The server can conduct an ECC point addition operation using (i) the derived first shared secret by the server and (ii) the received second shared secret from the key server. The resulting value from the ECC point addition operation can comprise a third shared secret. The server can input the third shared secret into a key derivation function in order to derive a symmetric ciphering key. The server can encrypt a random number generated by the server and a certificate for the server using the derived symmetric ciphering key and the symmetric ciphering algorithm. The server can generate a digital signature for a fourth message with the certificate and the random number using the private key corresponding to the public key in the certificate. The data encrypted by the server, including the digital signature, can comprise a first ciphertext. The server can send the device the fourth message, where the fourth message includes the server ephemeral public key and the first ciphertext.

The device can receive the fourth message from the server and take steps to process the message. The device can conduct a third ECDH key exchange with the received server ephemeral public key. The device, using the set of cryptographic parameters, can perform an elliptic curve point addition operation on (i) the received server ephemeral public key from the fourth message and (ii) the recorded network static public key. The device can input (x) the point derived from the ECC point addition and (y) the device ephemeral private key into an ECDH key exchange algorithm in order to mutually derive the third shared secret with the server. Device can input the third shared secret into a key derivation function in order to derive the same symmetric ciphering key derived and used by the server. The device can decrypt the first ciphertext using the derived symmetric ciphering key. The device can read the plaintext from the first ciphertext. The device can take additional steps to communicate with the server, such as (i) verifying a signature in a certificate within the first ciphertext, (ii) using the public key from the certificate to verify a server digital signature for the fourth message, (iii) recording and using a public key for the server from the certificate in the first ciphertext, and other possibilities exist as well. The device can then use the derived symmetric ciphering key to encrypt a

second ciphertext for the server and send the second ciphertext to the server in a fifth message. In exemplary embodiments, the derived symmetric ciphering key can comprise a first portion for encrypting and decrypting data sent from the server to the device and a second, different portion for encrypting and decrypting data sent from the device and to the server. The server can receive the fifth message and decrypt the second ciphertext using the same symmetric ciphering key derived by the server.

The systems and methods described above can also be used with particular implementations for a computing device and a server. A 5th generation or 6th generation wireless WAN network such as from 3GPP could utilize the steps above in order to conduct an ECDHE key exchange with a server authentication and a key server. For this embodiment, the computing device could comprise a wireless device or wireless terminal, including a mobile phone or smart phone. The server could comprise a “g Node B” for “next generation node b”, which provides equivalent functionality of a base transceiver station and manages the radio-frequency communications with the wireless device. The key server could comprise a secured server operating within the authentication function of a wireless network or associated with the authentication function of a wireless network for a mobile network operator. For the embodiment in this paragraph, the cryptographic parameters could comprise the values for curve 25519, although other ECC curves could be utilized as well.

The systems and methods described above can also be used with a device provisioning protocol. The computing device as described above can comprise an initiator according to the Device Provisioning Protocol specification version 1.0 from the WiFi Alliance ®. The server can comprise a responder according to the same specification. Subsequent versions of the specification can utilize the methods and systems described herein as well. The device can receive and record the network static public key in the form of a responder bootstrap public key. A key server could record the network static private key in the form of a responder bootstrap private key. The responder/server can receive the first message with (a) the device ephemeral public key from the initiator/device along with (b) a secure hash value of the responder bootstrap public key, and (c) an initial ciphertext. The responder/server can use the secure hash value of the responder bootstrap public key to select the key server for the device. The responder/server can forward the device ephemeral public key to the selected key server. The key server can conduct the second ECDH key exchange with the device ephemeral public key and the responder bootstrap private key and send the second shared secret to the responder/server. The server can use the second shared secret to decrypt the initial ciphertext received with the first message.

These as well as other aspects and advantages will become apparent to those of ordinary skill in the art by reading the following detailed description, with reference where appropriate to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Various exemplary embodiments are described herein with reference to the following drawings, wherein like numerals denote like entities.

5 Figure 1a is a graphical illustration of an exemplary system, where device communicates data with a network in order to conduct a key exchange, in accordance with exemplary embodiments;

 Figure 1b is a graphical illustration of hardware, firmware, and software components for a server, in accordance with exemplary embodiments;

10 Figure 1c is an illustration of exemplary network static public keys recorded by a plurality of devices, in accordance with exemplary embodiments;

 Figure 2a is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device, a server, and a key server, in accordance with exemplary embodiments;

15 Figure 2b is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive shared secrets, and for conducting a key derivation function using the derived shared secrets, in accordance with exemplary embodiments;

 Figure 2c is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key, and for using the derived shared secret key to encrypt and decrypt data, in accordance with exemplary embodiments;

20 Figure 2d is an illustration of an exemplary server database and an exemplary set of cryptographic parameters, in accordance with exemplary embodiments;

 Figure 2e is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key using ECC point multiplication, in accordance with exemplary embodiments;

25 Figure 3a is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a mobile device, a “g node b”, and a key server, in accordance with exemplary embodiments;

30 Figure 3b is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a client, a server, and a key server, in accordance with exemplary embodiments; and,

 Figure 3c is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by an initiator, a responder, and a key server, in accordance with exemplary embodiments.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS OF THE INVENTION**Figure 1a**

Figure 1a is a graphical illustration of an exemplary system, where device communicates data with a network in order to conduct a key exchange, in accordance with exemplary embodiments. The system 100 can include a device 103 and a network 105, where the nodes can communicate data 106 over an Internet Protocol (IP) network 107. Network 105 can comprise a plurality of servers supporting communication such as data 106 with a plurality of devices 103. In exemplary embodiments, network 105 can include a server 101 and a key server 102. The exemplary servers shown for network 105 in system 100 can be either different physical computers such as rack-mounted servers, or different logical or virtual servers or instances operating in a “cloud” configuration. Or, server 101 and key server 102 could represent different logical “server-side” processes within a network 105, including different programs running on a server that listen and communicate using different IP port numbers within one physical server. In exemplary embodiments, server 101 and key server 102 can operate using the physical electrical components depicted and described for a server 101 in Figure 1b below. Other possibilities exist as well for the physical embodiment of server 101 and key server 102 without departing from the scope of the present disclosure. In exemplary embodiments, server 101 can be described as a “first server” and key server 102 can be described as a “second server”. Further, the combination of a first server 101 and a second server 102 can comprise a network 105. The combination of a first server 101 and a second server 102 can also comprise a “set of servers”.

Although server 101 and key server 102 are depicted in Figure 1a as belonging to the same network 105, server 101 and key server 102 could be associated with different networks and communicate in a secure manner. Secure sessions between server 101 and key server 102 could be established over IP network 107 using methods including a physical wired connection via a local area network (LAN), transport layer security (TLS), a virtual private network (VPN), and IP Security (IPSEC), and other possibilities exist as well. As depicted in Figure 1a, server 101 and key server 102 could communicate over a private network 107a.

Device 103 can be a computing device for sending and receiving data. Device 103 can take several different embodiments, such as a general purpose personal computer, a mobile phone based on the Android ® from Google ® or the IOS operating system from Apple ®, a tablet, a networked device with a sensor or actuator for the “Internet of Things”, a module for “machine to machine” communications, a device that connects to a wireless or wired Local Area Network (LAN), an initiator according to the Device Provisioning Protocol specification (DPP) from the WiFi alliance, a router, and/or a server, and other possibilities exist as well without departing from the scope of the present

disclosure. Exemplary electrical components within a device 103 can be similar to the electrical components for a server 101 depicted and described in Figure 1b below, where device 103 can use electrical components with smaller capacities and lower overall power consumption, compared to the capacity and power consumption for the same electrical components in a server 101.

5 Device 103 can include a device identity 103i, which could comprise a string or number to uniquely identify device 103 with network 105 and/or server 101 and server 102. Device identity 103i could comprise a medium access control (MAC) address for a physical interface such as Ethernet or WiFi, a Subscription Permanent Identifier (SUPI) with 5G networks, an international mobile subscriber identity (IMSI) or international mobile equipment identity (IMEI) with 2G/3G/4G
10 networks, and other possibilities exist as well without departing from the scope of the present disclosure. In exemplary embodiments, device identity 103i can be written to hardware in device 103 and operate as a unique, long-term identity for device 103.

Device 103 can record at least one elliptic curve cryptography (ECC) static public key for network 105 comprising network static public key PK.network 102a. Network static public key 102a
15 could be recorded in nonvolatile or volatile memory within device 103. For embodiments where key 102a is recorded in nonvolatile memory, key 102a could be recorded by a device manufacturer or device distributor. For embodiments where key 102a is recorded in volatile memory, device 103 could obtain key 102a from a different server than server 101 for network 105 before sending data 106, such as device 103 obtaining key 102a via a secure session from a different server before sending data 106.
20 A device 103 can record a plurality of different network static public keys 102a in a network public key table 103t. Different keys 102a in a table 103t could be associated with different networks 105 or different servers 101 that device 103 communicates with over time. Exemplary data for a network public key table 103t for device 103 is depicted and described in connection with Figure 1c below. The different keys 102a can be associated with network names and/or Uniform Resource Locators (URLs) or domain names, such that device 103 can select the network static public key 102a based on
25 a URL or domain name for a network 105 or a server 101 where device 103 will send data 106.

Network static public key PK.network 102a can be obtained by device 103 before conducting an elliptic curve Diffie-Hellman (ECDH) key exchange or an ephemeral elliptic curve Diffie-Hellman (ECHDE) key exchange. Network static public key 102a could be obtained by device 103 in several
30 different ways. Network static public key 102a could be written into memory by a manufacturer, distributor, or owner of device 103 before device 103 connects with server 101 or a network 105. Network static public key 102a could be received by device 103 over an IP network 107 via a secured session, such as a TLS, DTLS, IPSec, or VPN connection before sending data 106 to server 101. In exemplary embodiments, network static public key 102a is recorded in device 103 in a secured and
35 authenticated manner, such that device 103 can trust network static public key 102a.

As one exemplary embodiment, network static public key 102a could be a public key within a certificate, where the public key 102a is signed by a certificate authority. Although not depicted in Figure 1a, device 103 could also record a certificate authority root certificate, and device 103 could (a) verify the signature of a certificate authority in a certificate for the public key 102a using (b) the
5 recoded root certificate for the certificate authority (and any intermediary parent certificates). Network static public key 102a could be processed or formatted according to a set of parameters 104, and network static public key 102a could also be compatible with parameters 104. Although public key 102a is described as “static”, the key could change over time such as with the expiration of a validity date when recorded in a certificate. Public key 102a could remain static over the period of
10 time for device 103 to conduct at least one ECDHE key exchange, where the ECDHE key exchange uses ephemeral or derived ECC PKI keys. Public key 102a could comprise a long-term public key for use by device 103 when communicating with network 105. Although the use of a certificate for public key 102a is described in this paragraph for public key 102a, the use of a certificate is not required. In an embodiment depicted in Figure 3c below, (i) public key 102a could comprise a responder bootstrap
15 public key and (ii) device 103 could comprise an initiator according to the DPP standard, which is also depicted and described in connection with Figure 3c below.

Cryptographic parameters 104 can specify values or settings for (i) conducting an ECDH or ECDHE key exchange, (ii) mutually deriving a symmetric ciphering key, and (iii) using a symmetric ciphering algorithm. As contemplated herein, cryptographic parameters 104 may also be referred to
20 as parameters 104. Each of device 103, server 101, and key server 102 can record at least one compatible subset of parameters within a set of cryptographic parameters 104. Parameters 104 can specify values for an elliptic curve cryptography (ECC) curve name, key length, key formatting (e.g. compressed or uncompressed), encoding rules, etc. As contemplated herein, the parameters 104 and cryptographic algorithms used with ECC PKI keys and a key exchange in the present disclosure can
25 be compatible and substantially conform with ECC algorithms and keys as specified in (i) the IETF Request for Comments (RFC) 6090 titled “Fundamental Elliptic Curve Cryptography Algorithms”, and (ii) IETF RFC 5915 titled “Elliptic Curve Private Key Structure”, and also subsequent and related versions of these standards. For use of ECC algorithms, parameters 104 can specify elliptic curve names such as, but not limited to NIST P-256, sect283k1, sect283r1, sect409k1, sect409r1, and other
30 possibilities exist as well. Further, elliptic curves that do not depend on curves currently specified by the National Institute of Standards and Technology (NIST) could be utilized as well, such as, but not limited to, Curve22519, curve448, or FourQ. Parameters 104 can specify domain parameters for nodes in system 100 to calculate values or numbers in a compatible manner, such as common base point G for use with ECC PKI key pairs and a defining equation for an elliptic curve. Other values within sets
35 of cryptographic parameters 104 are possible as well, without departing from the scope of the present disclosure. An exemplary set of cryptographic parameters 104 is depicted and described in connection

with Figure 2e below, and PKI keys used by device 103, server 101, and key server 102 could be associated with a member of the set of cryptographic parameters, such as a single row in the parameters 104 depicted in Figure 2e below.

Device 103 can include an ECC key pair generation algorithm 103x and server 101 can include
5 a compatible ECC key pair generation algorithm 101x. A key pair generation algorithm 103x or 101x can use (i) a random number generator in order to derive the ephemeral PKI private key and (ii) a selected set of cryptographic parameters 104 in order to calculate the ephemeral PKI public key. In exemplary embodiments, a random number for the ephemeral PKI private key multiplies the base point G from the parameters 104 in order to obtain the corresponding ephemeral PKI public key. Other
10 possibilities exist as well for the algorithms 103x and 101x to derive an ephemeral ECC PKI key pair without departing from the scope of the present disclosure. A key pair generation algorithm 103x for device 103 can output an ephemeral ECC PKI pair comprising device ephemeral public key Ed 103a and device ephemeral private key ed 103b. A key pair generation algorithm 101x for server 101 can
15 output an ephemeral ECC PKI pair comprising server ephemeral public key E1 101a and server ephemeral private key e1 101b. As contemplated in the present disclosure, the use of a capital letter as the first character for a PKI key can represent a public key, the use of a lower case letter as the first character for a PKI key can represent a private key. As contemplated in the present disclosure, the second letter for a PKI key can represent the entity the key is associated with or belongs to (e.g. “d” for device 103 and “1” for server 101).

Device 103 can also record a device static PKI key pair 103p in nonvolatile memory or within
20 a secure processing environment within device 103. The key pair 103p can be either (i) generated by device 103 during device manufacturing or device distribution, or (ii) generated externally from device 103 and written to device 103 in a secure manner during device manufacturing or device distribution. The PKI key pair 103p can comprise a device static private key d1 103d and a device static public key
25 D1 103c. The keys d1 103d and D1 103c could be formatted and compatible with the set of cryptographic parameters 104. In exemplary embodiments, public key D1 103c can be recorded in an X.509 certificate from a certificate authority.

As depicted in Figure 1a, server 101 can include a server identity 101i, a key pair generation
30 algorithm 101x, a set of cryptographic parameters 104, a server database 101d, and a server certificate 101c. Server identity 101i can comprise a name or number to uniquely identify server 101 in network 105 and/or IP network 107. In exemplary embodiments, server identity 101i can comprise a domain name service (DNS) name, which could comprise a string of characters and/or numbers. Server identity 101i could be associated with an IP address, such that the exemplary data 106 from device 103 could be routed to server 101 via the IP network 107. Server identity 101i could also comprise a
35 MAC address, and a server identity 101i could comprise multiple different values such as all of a MAC address, a DNS name, and virtual instance identity if server 101 operates as a virtual server. In

summary, server identity 101i can allow (a) a plurality of different devices 103 to (b) select and route data 106 to server 101 from a potential plurality of different servers and nodes. Server identity 101i could also comprise a server name indication (SNI) value. Other possibilities exist as well for the format, structure, or value for a server identity 101i without departing from the scope of the present disclosure.

A key pair generation algorithm 101x for server 101 was described above in connection with key pair generation algorithm 103x for device 103. Key pair generation algorithm 101x can derive ephemeral ECC PKI keys for server 101 to use with ECDHE key exchanges for a plurality of different devices 103. Note that although a single ECC PKI key pair of public key E1 101a and private key e1 101b is depicted for system 100, server 101 could derive and operate with a plurality of different keys E1 101a and e1 101b with different devices 103. The plurality of different keys E1 101a and e1 101b for communicating with different devices 103 could be recorded in a server database 101d as depicted and described in connection with Figure 2d below. The set of cryptographic parameters 104 for server 101 can be equivalent to or a superset of the cryptographic parameters 104 used by device 103. The description above for a set of parameters 104 used by a device 103 is also applicable to a set of parameters 104 used by a server 101.

Server database 101d for server 101 can comprise a database or memory recording data for server 101 to communicate with both a plurality of devices 103 and also at least one key server 102. An exemplary server database 101d is depicted and described in connection with Figure 2d below. Server database 101d can record values for PKI keys, derived shared secrets, derived symmetric ciphering keys, random numbers used in secure sessions, and related values in order to support the communications with both device 103 and key server 102. Server certificate 101c can comprise a certificate formatted according to the X.509 family of standards and include a static server 101 public key PK.S1 101p. Server certificate 101c can include a signature from a certificate authority for server public key PK.S1 101p. Although not depicted in Figure 1a, server 101 can also record and operate with a private key corresponding to public key PK.S1 101p.

As depicted in Figure 1a, key server 102 can include a key server identity 102i, a set of cryptographic parameters 104, a network static private key SK.network 102b, and a key server database 102d. Key Server identity 102i can comprise a name or number to uniquely identify key server 102 in network 105 and/or IP network 107. Key Server identity 102i can be similar to server identity 101i, except using a different value, name, or number in order to uniquely identify key server 102 within network 105. The set of cryptographic parameters 104 for server 102 can be equivalent to or a superset of the cryptographic parameters 104 used by device 103 and parameters 104 was also described above for device 103.

In exemplary embodiments, the parameters 104 used by both key server 102 and server 101 can be fully compatible, such as using the same ECC named curve, key lengths, encoding rules, etc.

Server database 102d for key server 102 can comprise a database or memory recording data for key server 102 to (i) communicate with a plurality of servers 101 and (ii) support server 101 communicating with a plurality of devices 103. Key server database 102d can be similar to server database 101d depicted in Figure 2d, except that key server database 102d can record values and data calculated by key server 102. Key server database 102d can record values for PKI keys, derived shared secrets, and related values in order to support the communications between (i) network 105 and/or server 101 and (ii) device 103. As depicted in Figure 1a, key server database 102d can record sets of data for different devices 103, where each set can comprise a row in a table with a device identity 103i, the network static public key value PK.network 102a, and the network static private key SK.network 102b. Although not depicted in Figure 1a, a key server database 102d could also record or store a secure hash value for the network public key 102a, where the algorithm for the secure hash value could be specified in a member of the set of cryptographic parameters 104. For some exemplary embodiments, (i) a device identity 103i could be omitted from a key server database 102d or (ii) the device identity 103i could comprise a secure hash value over either the network public key 102a or the device static public key 103c.

As depicted for a key server database 102d in Figure 1a, some devices 103 could share the same keys 102a and 102b, which could comprise shared keys 102z for the devices 103 as depicted and described in connection with Figure 1c below. Other devices 103 could record unique keys 102v, where devices 103 record a value for the network static public key PK.network 102a that is uniquely recorded in each device. A key server database 102d could record and track the associated network private and public keys for each device. In other exemplary embodiments, a key server 102 could omit recording device identities 103i in a database 102d, and key server 102 could associate and use a network static private key SK.network 102b with a particular server 101 (e.g. all data from a server 101 could use or be associated with the private key SK.network 102b).

Other possibilities exist as well for the mapping of network static private keys to either servers 101 or devices 103 without departing from the scope of the present disclosure. Also, although a single value for SK.network 102b is depicted as associated with a device 103, a key server 102 could also use multiple different values of SK.network 102b, such as (i) different values for SK.network 102b for different parameters 104 (e.g. different named curves), or (ii) separate values for SK.network 102b for digital signatures and ECDH key exchanges. In other words, a device 103 could also record the corresponding different multiple values for PK.network 102a, and select and use the public keys depending on requirements such as parameters 104 used or if the network public key will be used for verifying digital signatures or conducting ECDH key exchanges.

Key server 102 can record at least one network static private key SK.network 102b, which can be the private key corresponding to the network static public key PK.network 102a recorded by device 103 and described above for device 103. In exemplary embodiments and as depicted in Figure 1a and

also Figure 2a below, key server 102 may not communicate with device 103 directly, but rather communicates with server 101 through a private network 107a. Although not depicted in Figure 1a, a network 105 could operate a firewall in order to prevent packets or data from the public Internet (other than server 101) from reaching key server 102. In this manner by isolating key server 102 from IP network 107, security for the key server 102 and the network static private key SK.network 102b can be enhanced, since only authenticated and authorized nodes within network 105 and connected to private network 107a could communicate with server 102.

IP network 107 could be either a Local Area Network (LAN) or a Wide Area Network (WAN), or potentially a combination of both. IP network 107 could include data links supporting either IEEE 802.11 (WiFi) standards. Device 103 also utilize a variety of WAN wireless technologies to communicate data 106 with server 101, including Low Power Wide Area (LPWA) technology, 3rd Generation Partnership Project (3GPP) technology such as, but not limited to, 3G, 4G Long-Term Evolution (LTE), or 4G LTE Advanced, NarrowBand – Internet of Things (NB-IoT), LTE Cat M, proposed 5G networks, and other examples exist as well. Server 101 can connect to the IP network 107 via a wired connection such as, but not limited to, an Ethernet, a fiber optic, or a Universal Serial Bus (USB) connection (not shown). IP network 107 could also be a public or private network supporting Internet Engineering Task Force (IETF) standards such as, but not limited to, such as, RFC 786 (User Datagram Protocol), RFC 793 (Transmission Control Protocol), and related protocols including IPv6 or IPv4. A public IP network 107 could utilize globally routable IP addresses. Private IP network 107a could utilize private IP addresses which could also be referred to as an Intranet. Other possibilities for IP Network 107 and Private Network 107a exist as well without departing from the scope of the disclosure.

Figure 1b

Figure 1b is a graphical illustration of hardware, firmware, and software components for a server, in accordance with exemplary embodiments. Figure 1b is illustrated to include several components that can be common within a server 101. Server 101 may consist of multiple electrical components in order to communicate with a plurality of devices 101 and a key server 102. In exemplary embodiments and as depicted in Figure 1b, server 101 can include a server identity 101i, a processor 101e (depicted as “CPU 101e”), random access memory (RAM) 101f, an operating system (OS) 101g, storage memory 101h (depicted as “nonvolatile memory 101h”), a Wide Area Network (WAN) interface 101j, a LAN interface 101k, a system bus 101n, and a user interface (UI) 101m.

Server identity 101i could comprise a preferably unique alpha-numeric or hexadecimal identifier for server 101, such as an Ethernet MAC address, a domain name service (DNS) name, a Uniform Resource Locator (URL), an owner interface identifier in an IPv6 network, a serial number,

an IP address, or other sequence of digits to uniquely identify each of the many different possible nodes for a server 101 connected to an IP network 105. Server identity 101i could comprise a server name indicator (SNI). Server identity 101i can preferably be recorded in a non-volatile memory and recorded by a network 105 upon configuration of a server 101. Server identity 101i may also be a
5 number or string to identify an instance of server 101 running in a cloud or virtual networking environment. In exemplary embodiments, server 101 can operate with multiple different server identities 101i, such as a first server identity 101i comprising a DNS name and a second server identity 101i comprising an IP address and a port number. A different server 101 could be associated with a different IP address and port number for a network 105.

10 The CPU 101e can comprise a general purpose processor appropriate for higher processing power requirements for a server 101, and may operate with multiple different processor cores. CPU 101e can comprise a processor for server 101 such as an ARM® based process or an Intel® based processor such as belonging to the XEON® family of processors, and other possibilities exist as well. CPU 101e can utilize bus 101n to fetch instructions from RAM 101f and operate on the instruction.
15 CPU 101e can include components such as registers, accumulators, and logic elements to add, subtract, multiply, and divide numerical values and store or record the results in RAM 101f or storage memory 101h, and also write the values to an external interface such as WAN interface 101j and/or LAN interface 101k. In exemplary embodiments, CPU 101e can perform the mathematical calculations for a key pair generation step 101x and also an ECDH key exchange algorithm 220 depicted in Figure 2a,
20 Figure 2b, etc., below.

CPU 101e can also contain a secure processing environment (SPE) 101s in order to conduct elliptic curve cryptography (ECC) operations and algorithms, such as an ECC point addition step 214 depicted in Figure 2c below, as well as deriving ephemeral ECC PKI keys such as with key generation step 101x depicted and described in connection with Figure 1a above. SPE 101s can comprise a
25 dedicated area of silicon or transistors within CPU 101e in order to isolate the ECC operations from other programs or software operated by CPU 101e, including many processes or programs running operating system 101g. SPE 101s could contain RAM memory equivalent to RAM 101f and nonvolatile memory equivalent to storage memory 101h, as well as a separately functioning processor on a smaller scale than CPU 101e, such as possibly a dedicated processor core within CPU 101e. SPE
30 101s can comprise a “secure enclave” or a “secure environment”, based on the manufacturer of CPU 101e. In some exemplary embodiments, an SPE 101s can be omitted and the CPU 101e can conduct ECC operations or calculations without an SPE 101s.

RAM 101f may comprise a random access memory for server 101. RAM 101f can be a volatile memory providing rapid read/write memory access to CPU 101e. RAM 101f could be located on a
35 separate integrated circuit in server 101 or located within CPU 101e. The RAM 101f can include data recorded in server 101 for the operation when communicating with a plurality of devices 103 or a key

server 102. The system bus 101n may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures including a data bus. System bus 101n connects components within server 101 as illustrated in Figure 1b, such as transferring electrical signals between the components illustrated. Server 101 can include multiple different versions of bus 101n to connect different components, including a first system bus 101n between CPU 101e and RAM 101f (which could be a memory bus), and a second system bus 101n between CPU 101e and WAN interface 101j or LAN interface 101k, which could be an I2C bus, an SPI bus, a PCI bus, or similar data busses.

In exemplar embodiments, RAM 101f operating with server 101 can record values and algorithmic steps or computer instructions for conducting an ECDH key exchange, including a key pair generation step 101x, a secret X1 211a (depicted in Figure 2b below) and also a secret X2 212a (depicted in Figure 2b below). The depicted values and algorithms can be recorded in RAM 101f so that CPU 101e can conduct ECC operations and calculations quickly using the values. The depicted values could also be recorded in other locations for longer-term or nonvolatile storage, such as within a server database 101d. Additional or other values besides the ones depicted in Figure 1b can also be recorded in RAM 101f in order to support server 101 conducting the communications, steps, and message flows depicted in Figure 2a below and other Figures herein.

The operating system (OS) 101g can include Internet protocol stacks such as a User Datagram Protocol (UDP) stack, Transmission Control Protocol (TCP) stack, a domain name system (DNS) stack, a TLS stack, a DPP stack, etc. The operating system 101g may include timers and schedulers for managing the access of software to hardware resources within server 101, where the hardware resources managed by OS 101g can include CPU 101e, RAM 101f, nonvolatile memory 101h, and system bus 101n, and well as connections to the IP network 107 via a WAN interface 101j. The operating system shown of 101g can be appropriate for a higher power computing device with more memory and CPU resources (compared to a device 103). Example operating systems 101g for a server 101 includes Linux or Windows® Server, and other possibilities exist as well. Although depicted as a separate element within server 101 in Figure 1b, OS 101g may reside in RAM 101f and/or nonvolatile memory 101h during operation of server 101.

As depicted in Figure 1b, OS 101g in Figure 1b can contain algorithms, programs, or computer executable instructions (by processor 101e or SPE 101s) for an ECDH key exchange algorithm 220 (depicted and described in Figure 2b and Figure 2e below), a key derivation function (KDF) 216 (depicted and described in Figure 2b and Figure 2e below), and also an ECC point addition operation 214 (depicted and described in Figure 2b and Figure 2e below). The algorithms could be included either (i) within the kernel of OS 101g, or (ii) as a separate program or process loaded by OS 101g and operated by OS 101g. OS 101g can also read and write data to a secure processing environment SPE 101s, if CPU 101e contains SPE 101s.

Nonvolatile memory 101h or “storage memory” 101h (which can also be referred to herein as “memory 101h”) within server 101 can comprise a non-volatile memory for long-term storage of data, including times when server 101 may be powered off. Memory 101h may be a NAND flash memory or a NOR flash memory and record firmware for server 101, such as a bootloader program and OS 101g. Memory 101h can record long-term and non-volatile storage of data or files for server 101. In an exemplary embodiment, OS 101g is recorded in memory 101h when server 101 is powered off, and portions of memory 101h are moved by CPU 101e into RAM 101f when server 101 powers on. Memory 101h (i) can be integrated with CPU 101e into a single integrated circuit (potentially as a “system on a chip”), or (ii) operate as a separate integrated circuit or a removable card or “disk”, such as a solid state drive (SSD). Storage memory 101h can also comprise a plurality of spinning hard disk drives in a redundant array of independent disks (RAID) configuration. Memory 101h may also be referred to as “server storage” and can include exemplary file systems of FAT16, FAT 32, NTFS, ext3, ext4, UDF, or similar file systems. As contemplated herein, the terms “memory 101h”, “storage memory 101h”, and “nonvolatile memory 101h” can be considered equivalent.

As depicted in Figure 1b, non-volatile memory 101h can record a server database 101d, a device static public key D1 103c, and cryptographic parameters 104. Exemplary data within a server database 101d is depicted and described in connection with Figure 2d below. Although depicted in Figure 1b as recorded within memory 101h, a server database 101d could also operate as a separate server than server 101 in a network 105, and server 101 could query the server database 101d using a private network 107a. The device static public key D1 101c could be received by server 101 from a device manufacturer or a device owner, or directly from device 103 through IP network 107. In addition, as depicted in Figure 1b, memory 101h can record the parameters 104 which were depicted and described in connection with Figure 1a above and also Figure 2d below.

Server 101 can include a WAN interface 101j to communicate with IP network 107 and a plurality of devices 103, as depicted in Figure 1a above (where Figure 1a depicts a single device 103). WAN interface 101j can comprise either a wired connection such as Ethernet or a wireless connection. For wireless configurations of server 101, then WAN interface 101j can comprise a radio, which could connect with an antenna in order to transmit and receive radio frequency signals. For a wireless configuration of server 101, WAN interface 101j within server 101 can provide connectivity to an IP network 107 through 3GPP standards such as 3G, 4G, 4G LTE, and 5G networks, or subsequent and similar standards. In some exemplary embodiments, server 101 can comprise a “g node b” or gNb in a 5G network (or equivalent functionality in 6G or subsequent networks), and WAN interface 101j can comprise a 5G radio access network (RAN) interface. WAN interface 101j can also comprise a wired connection such as digital subscriber line (DSL), coaxial cable connection, or fiber optic connection, and other possibilities exist as well without departing from the scope of the present disclosure.

Server 101 may also operate a LAN interface 101k, where LAN interface 101k can be used to connect and communicate with other servers in a network 107, such as key server 102 through private network 107a. LAN interface 101k can comprise a physical interface connected to system bus 101n for server 101. In exemplary embodiments, LAN interface 101k can comprise an Ethernet or fiber
5 optic wired connection. In other words, (i) LAN interface 101k can connect server 101 to private network 107a (which could comprise an IP network with private IP addresses that are not globally routable), and (ii) WAN interface 101j can comprise an interface for communicating with a plurality of devices 103 through insecure networks such as the globally routable public Internet. The use of a separate WAN interface 101j and LAN interface 101k can increase the security of operation for server
10 101. However, the use of separate physical interfaces for LAN interface 101k and WAN interface 101j can be omitted, and a single physical interface such as Ethernet or fiber-optic could be used by server 101 to communicate with both devices 103 and key server 102.

Server 101 may also optionally include user interface 101m which may include one or more sub-servers for receiving inputs and/or one or more sub-servers for conveying outputs. User interfaces
15 are known in the art and may be simple for many servers 101 such as a few LED lights or and LCD display, and thus user interfaces are not described in detail here. User interface 101m could comprise a touch screen or screen display with keyboard and mouse, if server 101 has sophisticated interaction with a user, such as a network administrator. Server 101 can optionally omit a user interface 101m, if no user input or display is required for establishing communications within a network 105 and/or IP
20 network 107. Although not depicted in Figure 1b, server 101 can include other components to support operation, such as a clock, power source or connection, antennas, etc. Other possibilities exist as well for hardware and electrical components operating in a server 101 without departing from the scope of the present disclosure. Using the electrical components depicted in Figure 1b, a server 101 could send and receive the data 106 in Figure 1a in an encrypted and secure manner after conducting the
25 authenticated ECDHE key exchange as contemplated herein, in order to derive a symmetric ciphering key to encrypt and decrypt messages within data 106 with a plurality of devices 103.

Although not depicted in Figure 1b, devices 103 such as the device 103 depicted in Figure 1a above can include (a) equivalent internal electrical components depicted for a server 101 in order to
30 (b) operate as devices 103. A device 103 in Figure 1a could include a processor similar to CPU 101e, with primary differences for the processor in a device being reduced speed, a smaller memory cache, a smaller number and size of registers, with an exemplary use of 32 bits for datapath widths, integer sizes, and memory address widths, etc., for a device 103. In contrast, an exemplary 64 bit datapaths could be used for CPU 101e in server 101 (although device 103 could also use 64 bit wide datapath widths if device 103 comprises a mobile phone such as a smart phone). For embodiments where
35 device 103 comprises a transducer device for sending and receiving transducer data with a network

105, then a CPU in device 103 could comprise an exemplary 32 bit processor, although other possibilities exist as well.

Similarly, RAM in a device 103 could be a RAM similar to RAM 101f in server 101, except the RAM in a device 103 could have fewer memory cells such as supporting exemplary values less than or equal to an exemplary 4 gigabytes, while RAM 103f in server 101 could support more memory cells such as greater than or equal to an exemplary 8 gigabytes. In exemplary embodiments, the electrical and physical components of a key server can be equivalent to the electrical components for a server 101 in Figure 1b, with different data recorded in RAM 101f for a key server 102, as well as different data recorded in memory 101h for a key server 102. For example, a key server 102 could record the network static private key SK.network 102b in memory 101h, which could include secure disk storage using disk or file encryption.

Figure 1c

Figure 1c is an illustration of exemplary network static public keys recorded by a plurality of devices, in accordance with exemplary embodiments. Figure 1c depicts PKI keys recorded for an exemplary three different devices 103, although a system 100 and other systems herein could operate with potentially millions or more devices 103. The data depicted for each device in Figure 1c can comprise exemplary data for a network public key table 103t for a device 103, which is also depicted and described in connection with Figure 1a above. The exemplary values recorded for network static public keys depicts different embodiments where both (i) a device 103 can record a network static public key PK.network 102a that is shared with other devices 103, and (ii) the network static public key PK.network 102a recorded by device 103 could be unique for device 103 (e.g. not shared with other devices 103 in a system 100 above or a system 200 below, as well as other systems herein). A network public key table 103t for device 103 can record values of a key identity, a network name for network 105, an identity for server 101 comprising ID.server 101i, and also a value for the network static public key PK.network 102a. As depicted in Figure 1c, a device 103 can record multiple different values for use with multiple different networks 105 and/or servers 101.

The first two entries for network static public keys PK.network 102a for a first device 103 (1) and a second device 103 (2) in Figure 1c depicts the same alphanumeric values for base91 binary to text encoding for an exemplary network static public keys PK.network 102a in a first device 103 (1) and a second device 103 (2), where the key value is depicted for a network 105 of "Network A". Likewise, the second two entries for network static public keys PK.network 102a for a first device 103 (1) and a second device 103 (2) in Figure 1c depicts the same alphanumeric values for base91 binary to text encoding for an exemplary network static public key PK.network 102a in a first device 103 (1)

and a second device 103 (2). Note that although a single value is depicted for PKI keys in a network public key table 103t, the values or numbers for keys recorded could comprise a point on an ECC curve with both an X coordinate and a Y coordinate. For illustration purposes in Figure 1c, only the X coordinate is displayed and the Y coordinate could be calculated from the X coordinate using the equation for an ECC curve in a set of cryptographic parameters 104a for the PKI keys.

The depiction of these keys PK.network 102a illustrates the use of shared keys 102z for a plurality of different devices 103. Although only two devices are depicted with shared keys 102z, many more devices could also record the same shared keys for PK.network 102a. Each of the shared keys 102z is associated with a different network 105, identified with an exemplary different network name. In this manner, a plurality of different devices 103 can record and use the same value for a network static public key PK.network 102a. As described above, the value in a table 103t including network static public key PK.network 102a could be written in device before the device sends the first message 203 in Figure 2a below. The data could be recorded by a device manufacturer, a device distributor, or a device owner, and other possibilities exist as well for the source of PK.network 102a without departing from the scope of the present disclosure.

The same values for shared keys 102z across different devices 103 could be recorded in device 103 during manufacturing or before distribution to end users of device 103. In this manner, devices 103 could be received by end users in a “partially configured” yet secure state, such that a device 103 could use the recorded keys PK.network 102a with a server 101 and/or network 105, where a server 101 does not operate or record the corresponding network static private key SK.network 102b. As depicted and described in connection with Figures 2a, 2b, etc. below, a key server 102 could record and operate with the corresponding network static private key SK.network 102b and thus the key SK.network 102b can remain secured and not distributed out or sent to a server 101. In this manner, encrypted communications for data 106 in Figure 1a can be transferred between device 103 and server 101 without server 101 recording the key SK.network 102b. This increases the security of a system 100 and other systems herein, because server 101 may be exposed to an IP network 107 while key server 102 recording the SK.network 102b can be connected to a private network 107a.

By using a set of shared keys 102z across a plurality of device 103, a key server 102 or a network 105 can control access of the devices 103 as a group. For example, a network 105 could deny access to the private key corresponding to the public key for the first depicted value of PK.network 102a in a first device 103 (1). That action by network 105 would also deny a second device 103 (2) access to the private key corresponding to the public key for the first depicted value of PK.network 102a in the second device 103 (2). In this manner, network 105 could control access to a plurality of different devices 103 by controlling access to a single value of SK.network 102b, where (i) the plurality of different devices 103 record the corresponding PK.network 102a as shared keys 102z. Other benefits for using shared keys 102z can be available as well, such as simplifying manufacturing

or distribution, since the same key value for PK.network 102a could be recorded with multiple different devices 103. In other words, a device manufacturer or device distributor would not need to keep track of which value for PK.network 102a belongs with which device 103 for embodiments where shared keys 102z are utilized. However, the use of shared keys 102z for multiple different devices 103 is not required for some exemplary embodiments.

In exemplary embodiments, network static public keys PK.network 102a can also comprise a unique key for each device 103 in a system 100 and other systems herein. Thus, some exemplary embodiments also support the use of a network static public key PK.network 102a that is not shared across multiple different devices 103. For these exemplary embodiments, and as depicted in Figure 1c, a device 103 can record a unique key 102v (depicted as “Per Device or Unique Network Static Public Keys 102v” in Figure 1c). For example, the depicted value for the third key for device 103 (1), (2), and (3) in Figure 1c is shown as unique for each device. A key server 102 could also record the corresponding network static private key SK.network 102b that is unique for each device in a key server database 102d as depicted for unique keys 102v in Figure 1a. In this manner, a network 105 can control access to server 101 and/or network 105 on a per-device basis using the unique key 102v. For example, key server 102 could deny access to device 103 (3) (while continuing to allow service for device 103 (1) and 103 (2)), by denying access or cryptographic operations with the secret key SK.network 102b in a key server 102 corresponding to the public key PK.network 102a recorded by device 103 (3). Other benefits for recording network static public keys PK.network 102a as unique keys 102v for devices 103 exist as well without departing from the scope of the present disclosure.

Although not depicted in Figure 1c, each row or network static public key PK.network 102a could also be stored with a set of cryptographic parameters 104a, such as specifying an ECC named curve associated with the public key 102a. A network 105 or a server ID 101i could be associated with multiple different network static public keys PK.network 102a, where the different keys 102a for the same network 105 or server ID 101i are associated with different parameters 104a. Although depicted as alphanumeric values for the network static public key PK.network 102a, a network public key table 103t could store the public key 102a as separate certificates for the public keys. In addition, a network public key table 103t could store a secure hash value for the network static public key PK.network 102a, where the secure hash algorithm 104d for the secure hash value could be specified by parameters 104, as depicted and described in connection with Figure 2d below. In addition, a table 103t could include a key server identity 102i associated with the network static public key PK.network 102a.

Figure 2a

Figure 2a is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device, a server, and a key server, in accordance with exemplary embodiments. System 200 can include a device 103, server 101, and a key server 102. Device 103 was depicted and described in connection with Figure 1a, and Figure 1c above. Server 101 and key server 102 were depicted and described in connection with Figure 1a above, and server 101 was depicted and described in connection with Figure 1b above. Server 101 can record and operate a server database 101d, and key server 102 can record and operate a database 102d. Individual steps and components used in system 200 in Figure 2a are also additionally depicted and described in subsequent Figures 2b, 2c, and 2d. Before starting the steps and message flows depicted in Figure 2a, device 103 can securely receive and record a network public key PK.network 102a, which was also depicted and described in connection with Figure 1a and Figure 1c. The corresponding private key for PK.network 102a can be securely recorded in key server 102 within network 105 as SK.network 102b.

For system 200, server 101 and key server 102 may establish a secure session 221, which could comprise establishing a secure communications link between the two servers using protocols such as TLS, IPSec, a virtual private network (VPN), a secure shell (SSH), or similar networking, transport, or application layer technologies in order to establish secure communications between key server 102 and server 101. Secure session 221 can utilize certificates for the two servers in order to provide mutual authentication and mutual key derivation for a symmetric encryption key in secure session 221. Secure session 221 can also be conducted over private network 107a, although the secure session 221 could also be established or conducted through an IP network 107 such as the globally routable Public Internet. Other possibilities exist as well for establishing a secure session 221 between server 101 and key server 102 without departing from the scope of the present disclosure. Although not depicted in Figure 2a, firewalls between server 101 and key server 102 could also be utilized in order to establish or conduct secure session 221. At step 201b, server 101 can begin listening for incoming messages from a device 103 using a physical network interface such as WAN interface 101j that provides connectivity to the IP network 107 and server 101 can use a specific port number such as, but not limited to, TCP port 443 to listen for incoming data 106 from a device 103.

At step 201a, device 103 can be powered on and begin operating, in order to establish connectivity with an IP network 107. At step 202, device 103 can read an address for server 101 from memory or a network public key table 103t, and the address can comprise a DNS name or an IP address for server 101. The DNS name or IP address for server 101 could be recorded or received along with the key PK.network 102a, or device 103 could conduct a DNS query to obtain the address. At step 202, device 103 can also read the set of cryptographic parameters 104 and select a subset of the cryptographic parameters 104a in order to establish communications with server 101. An exemplary

subset of cryptographic parameters 104a in a step 202 can comprise a member of the set the cryptographic parameters 104 depicted and described in connection with Figure 2d below (e.g. one line of values in cryptographic parameters 104 in Figure 2d below). In step 202, device 103 can select a subset of cryptographic parameters 104a that is compatible with PK.network 102a. The subset of
5 cryptographic parameters 104a that are compatible with PK.network 102a could also be recorded in nonvolatile memory in device 103 along with network public key PK.network 102a at the time PK.network 102a was recorded or received by device 103.

A step 202 can also comprise device 103 also using a random number generator in order to output a random number 202a for use in subsequent communications with server 101. Although the
10 term “random number” is described herein, a random number could comprise a pseudo random number processed by device 103 using information entropy available to device 103. The random number 202a processed in a step 202 could contain the number of bits specified by a selected subset of cryptographic parameters 104, such as a random length 104g. Random number 202a generated or derived by a device 103 in a step 202 could also comprise a “number used once” (nonce).

Device 103 can then conduct a key pair generation step 103x as depicted and described in connection with Figure 1a above using the selected subset of cryptographic parameters 104a. The parameters 104 could specify a named curve and parameters to derive a device ephemeral private key ed 103b and a device ephemeral public key Ed 103a. The device ephemeral private key ed 103b can
15 comprise a random number generated using a random number generator. The device ephemeral public key Ed 103a could be derived using (i) ECC point multiplication from a base point G for a named curve within cryptographic parameters 104a and (ii) the device ephemeral private key ed 103b. Other possibilities exist as well for the steps a device 103 can use in a key pair generation step 103x without departing from the scope of the present disclosure.

Device 103 can then use (i) the recorded address for server 101 (possibly from a table 103t) and (ii) connectivity to IP network 107 from step 202 to send a message 203 to server 101. Message
25 203 and other messages contemplated herein can be sent as either TCP or UDP messages, and other possibilities exist as well for the formatting and transfer of messages without departing from the scope of the present disclosure. In exemplary embodiments, device 103 both uses an IP address and port number as a source IP address and port to send message 203 to server 101 and then also the same IP address and port number to listen for responses or messages from server 101. In this manner, device
30 103 can send a message 203 and receive a response message 206c below through an IP network 107, where intermediate nodes on the IP network 107 may conduct network address translation (NAT) routing.

Message 203 can include the random number random1 202a from a step 202, the device
35 ephemeral public key Ed 103a, and the subset of cryptographic parameters 104a. Message 203 may also optionally include a device identity of ID.device 103i, but the device identity of ID.device 103i

can also be omitted from a message 203 in some exemplary embodiments. For embodiments where message 203 optionally excluded device identity ID.device 103i, then an identity for device 103i can optionally be transmitted in later messages. Omitting ID.device 103i from message 203 can increase security for message 203 since an identity for device 103 would not be sent as plaintext in a message 203. Although not depicted in Figure 2a, message 203 could also optionally include an identity for key server 102 comprising ID.key-server 102i, such that server 101 can determine which key server 102i should be associated with message 203. Note that an identity for key server 102 of ID.key-server 102i can be omitted from a message 203, and server 101 can select a key server 102 from other means in a step 205b below.

As depicted in Figure 2a, message 203 could also optionally include a secure hash value 250 (also depicted in Figure 2d below) such as, but not limited to, SHA-256 of the network static public key PK.network 102a. Device 103 can send the hash value 250 of key 102a to server 101 in a message 203, in order for server 101 to identify which of a plurality of possible key servers 102 could be used to process data within message 203, which is further described for a step 205b below. For embodiments where a secure hash value 250 of key 102a is included in a message 203, then the message 203 could optionally exclude the selected subset of cryptographic parameters 104a associated with keys PK.network 102a and Ed 103a. For other embodiments, a key identity for key 102a could be selected by device 103 from a table 103t and the key identity for key 102a could be sent in a message 203 instead of a hash value 205 for key 102a. A server 101 and key server 102 could store the key identity for key 102a and select the key 102a using the key identity for key 102a.

Server 101 receiving the message 203 with the hash value 250 could determine the set of parameters 104a to use for key Ed 103a based on the hash value 250. For example, and as depicted in Figure 2d below, a server database 101d could maintain mapping of hash values 250 and parameters 104a, and server 101 could conduct a query of database 101d using the received hash value 250 in order to select the parameters 104a for further processing and cryptographic operations with key Ed 103a. Or, in an exemplary embodiment cryptographic parameters 104a as transmitted via an IP network 107 or private network 107a could comprise the secure hash 250 of key 102a, where the secure hash 250 of key 102a can specify which subset of a set of cryptographic parameters 104 to utilize for ECC operations (in other words the subset of parameters 104 can comprise parameters 104a). For embodiments where device 103 uses a unique key 102y, then the secure hash value 250 can also comprise a device identity 103i (since the secure hash value 250 would be unique for device 103). Secure hash value 250 could also be omitted from message 203 in some exemplary embodiments.

Server 101 can receive message 203 and begin conducting steps in order to process the message. At step 204, server 101 can read the subset of cryptographic parameters 104a in the message 203 and being using the subset of cryptographic parameters. Or, for embodiments that include hash

value 250, then parameters 104a could be omitted from message 203 and server 101 could select the parameters 104a from a server database 101d using the hash value 205, such as with the exemplary server database depicted in Figure 2d below. At step 204, server 101 can comprise a public key validation step on received device ephemeral public key Ed 103a in order to ensure the key is valid and on the selected curve in parameters 104a. Step 204 by server 101 can comprise conducting the steps for an ECC Full Public-Key Validation Routine in section 5.6.2.3.2 of FIPS publication SP 800-56A (revision 2) for the received device ephemeral public key Ed 103a. Alternatively, step 204 can comprise server 101 performing the steps ECC Partial Public-Key Validation Routine in section 5.6.2.3.3 of the same FIPS publication. Other example steps within a public key validation step 204 can comprise (i) verifying the public key is not at the “point of infinity”, and (ii) verifying the coordinates of the point for the public key are in the range $[0, p-1]$, where p is the prime defining the finite field. Other possibilities exist as well for evaluating and validating a received public key is cryptographically secure in a public key validation step 204, without departing from the scope of the present disclosure. As contemplated in the present disclosure a device 103, server 101, and key server 102 can conduct a public key validation step 204 each time a public key or a point on an elliptic curve is received.

At step 205a and after a key validation step 204, server 101 can record the data received from the message 203 in a server database 101d. Exemplary values and data for a server database 101d are depicted and described in connection with Figure 2d below. At step 205a, server 101 can record in server database 101d the values of random number 202a, device ephemeral public key Ed 103a, and the subset of cryptographic parameters 104d. For embodiments where device identity ID.device 103i is also received in message 203, then server 101 can also record device identity ID.device 103i in server database 101d. A step 205a can also include (i) storing both Ed 101a and random 202a in database 101d, and (ii) confirming that Ed 101a and random 202a are not reused. Security of a system 200 and system 100 and other systems herein can be increased through prohibiting the reuse of ephemeral PKI key pairs and also random numbers. If numbers or keys are reused, then server 101 could respond with a request for device 103 to generate a new ephemeral PKI key pair and/or random number 202a before proceeding to further steps. For embodiments requiring higher security, then hash values for received keys Ed 101a could be stored in a database 101d (instead of the value of key Ed 101a), and a new Ed 101a received by server 101 could be determined as new or reused by calculating a hash value for the received key Ed 101a and comparing with stored values for Ed 101a.

At step 205a, server 103 can also record the originating source IP address and port number 203a (depicted in Figure 2d below) for message 203, in order to subsequently transmit a message 206c below back to the same IP address and port number. In this manner, message 206c below can be routed by intermediate nodes on IP network 107 back to the source IP address and port number used by device 103 to transmit message 203. In other words, (i) the destination IP address and port number

of a subsequent message 206c from server 101 to device 103 can comprise the source IP address and port number 203a (depicted in Figure 2d below) received in message 203, and (ii) the source IP address and port number 203a (depicted in Figure 2d below) from message 203 can be recorded in a server database 101d. In this manner, device 103 can be tracked or identified by server 101 during the brief
5 period of time of the message flows in Figure 2a using the source IP address and port number from message 203 for embodiments where device identity ID.device 103i is not included in message 203. A step 205a can also comprise server 101 generating a second random number 205r using parameters 104a for use in subsequent messages with device 103. The first random number can comprise random number random1 202a derived by device 103.

10 At step 205b, server 101 can select key server 102 for subsequent communications and processing of the received device ephemeral public key Ed 103a. Note that a system 100 could comprise both a plurality of devices 103 and a plurality of key servers 102. In exemplary embodiments server 101 should select in step 205b the proper key server 102 for conducting subsequent steps in Figure 2a. In other words, without data or values from a message 203, server 101 may know which
15 of a possible plurality of key server 102 may record the network static private key SK.network 102b for use with or associated with device ephemeral public key Ed 103a. Server 101 could use one of several possible methods for selecting key server 102 in a step 205b, including a combination of the following embodiments.

A first embodiment for selecting key server 102 in a step 205b could comprise server 101
20 selecting the same key server 102 for all keys Ed 103a from all devices 103. For example for this first method, server 101 could listen or operate on (i) a specific IP address and port number or (ii) with a specific DNS name or server name indicator (SNI) in step 201b, where the use of (i) or (ii) could be specified or associated with network static public key PK.network 102a. As mentioned above for a step 201a, device 103 can select the address of server 101 using the server address of server 101
25 recorded with PK.network 102a (possibly from a table 103t in Figure 1c). Server 101 could determine that all messages 203 received using (i) or (ii) are associated with a specific key server 102. For this first embodiment of a step 205b, a plurality of devices 103 could store shared keys 102v for PK.network 102a, as depicted and described in connection with Figure 1c.

A second embodiment of a step 205b for selecting key server 102 of received device ephemeral
30 public key Ed 103a could comprise using an identity of key server 102 in a message 203 from device 103. As described above for a message 203, the message 203 can optionally include an identity for key server 102 comprising ID.key-server 102i. For these embodiments, server 101 can select the key server 102 using the ID.key-server 102i in message 203. A third embodiment for a step 205b of selecting key server 102 for received device ephemeral public key Ed 103a could comprise using an
35 identity of device 103 in a message 203 comprising ID.device 103i. As described above for a message 203, the message 203 can optionally include an identity for device 103, and server 101 using database

101d could include a table to map ID.device 103i to key server 102. For this third embodiment of a step 205b, server 101 could conduct a query of server database 101d to select the key server 102 for device 103 using ID.device 103i.

5 A fourth embodiment for a step 205b to select a key server 102 for received device ephemeral public key Ed 103a could comprise using the subset of cryptographic parameters 104a in a message 203 from device 103. Server 101 could record that a first subset of cryptographic parameters 104a are associated with a first key server 102, and a second subset of cryptographic parameters 104a are associated with a second key server 102, etc. A fifth embodiment for a step 205b to select a key server 102 for received device ephemeral public key Ed 103a could comprise message 205 including a secure
10 hash value 250 (in Figure 2d) of network static public key PK.network 102a, and server 101 with database 103d could include a table to map the secure hash value 250 of PK.network 102a to key server 102. Other possibilities exist as well for server 101 to conduct a step 205b to select a key server 102 using data in a message 203 without departing from the scope of the present disclosure. For embodiments of step 205b, the selection of key server 102 can comprise the selection of an identity
15 for key server 102 of key server identity 102i, and subsequent data such as message 206a could be sent or routed through IP network 107a using the key server identity 102i.

After selecting key server 102 in a step 205b, server 101 can then send key server 102 a message 206a through the secure session 221. Message 206a can include an identity for server 101 comprising ID.server 101i, the received device ephemeral public key Ed 103a, and the subset of
20 cryptographic parameters 104a. For embodiments where device identity ID.device 103i was included in a message 203, then ID.device 103i could be included in a message 206a as well. However, device identity ID.device 103i could be omitted from a message 203 and for these embodiments then message 206a can exclude device identity ID.device 103i as well. Server identity ID.server 103i can be useful for communications between key server 102 and server 101 for a system 100 and system 200, since
25 either (i) server 101 may communicate with a plurality of different key servers 102, and/or (ii) key server 102 may communicate with a plurality of different servers 101.

Server 101 can then conduct a key pair generation step 101x as depicted and described in connection with Figure 1a above using the selected subset of cryptographic parameters 104a. The parameters 104a could specify a named curve and parameters to derive a server ephemeral private key
30 e1 101b and a server ephemeral public key E1 101a. The server ephemeral private key e1 101b can comprise a random number generated using a random number generator. The server ephemeral public key E1 101a could be derived using (i) ECC point multiplication from a base point G for a named curve within cryptographic parameters 104a and (ii) the server ephemeral private key e1 101b. Although message 206a is depicted in Figure 2a as transmitted or sent by server 101 to key server 102
35 before server 101 derives ephemeral server PKI keys in a step 101x, a message 206a could be sent by server 101 after server 101 conducts the step 101x. Key pair generation step 101x can also confirm

that the server ephemeral PKI key pair for server 101 is not reused, such as storing hash values for public keys E1 101a in a database 101d and then comparing the hash value for a new key E1 101a from a step 101x with the stored hash values. If the derived new key E1 101a matches a stored hash value 101a from a database 101d, then the new key E1 101a could be discarded and a different key E1 101a derived.

Key server 102 can receive the message 206a via the secure session 221 and conduct a series of steps to process the message and respond. A first step conducted by key server 102 can comprise a key validation step 204, where the key validation step 204 conducted by key server 102 can be equivalent or compatible with the key validation step 204 conducted by a server 101 as described above. For a key validation step 204, a node can reply with a failure or reject message if the key validation step 204 fails, such as if a received ECC public key fails to fall on the named elliptic curve as specified by a subset of cryptographic parameters 104a.

At step 205c, key server 102 can use data from message 206a in order to select a network static private key SK.network 102b for subsequent steps such as a step 211. For embodiments where message 206a includes either (i) an identity for device 103 such as ID.device 103i, or (ii) identifying information for SK.network 102b for key server 102 to utilize (such as hash 250 of the public key PK.network 102a for SK.network 102b), then key server 102 could use the identifying information in message 206a to select the network static private key SK.network 102b from a key server database 102d, where an exemplary key server database 102d is depicted and described in connection with in Figure 1a above. For some exemplary embodiments, the key server database 102d can record a network static private key SK.network 102b for each set of cryptographic parameters 104a, and subsequently select the key 102b using the parameters 104a received in a message 206a. In other words, an identity for device 103 or hash 250 of PK.network 102a could be omitted, and a key server 102 could use a step 205c to select a network static private key SK.network 102b using a set of cryptographic parameters 104a.

Key server 102 can then conduct an ECDH key exchange step 211 using (i) the selected network static private key SK.network 102b, (ii) the received device ephemeral public key Ed 103a, and (iii) the set of cryptographic parameters 104a. Exemplary details for an ECDH key exchange step 211 are depicted and described in connection with Figure 2b below. The output of an ECDH key exchange step 211 can comprise point X1 211a.

Key server 102 can then send server 101 a message 206b, where the message 206b includes point X1 211a, as well as an identity for key server 102 comprising ID.key-server 102i and cryptographic parameters 104a associated with point X1 211a. Message 206b can be transmitted through secure session 221. If device identity 103i or other identifying information such as hash 250 was included in message 206a, then message 206b could also include device identity 103i or the other identifying information for a device 103. Or, both message 206a and message 206b can include a

transaction identity or session identity, such that server 101 can associate the received value X1 211a with a received device ephemeral public key Ed 103a.

Server 101 can receive message 206a with point X1 211a and conduct a series of steps in order to derive a mutually shared and authenticated key exchange with device 103. As contemplated herein, the authentication performed by server 101 can comprise a “one-way” authentication with device 103. Authentication of server 101 or network 105 can be provided by the depicted key exchange with steps 211 and 213, since network 105 from system 100 with both server 101 and key server 102 conducts an ECDH key exchange using at least, in part, the network static private key SK.network 102b. The “one-way” authentication from the ECDH key exchange is also not completed until both sides have successfully used a symmetric ciphering key derived from the ECDH key exchange. In other words, a device that successfully mutually derives a symmetric ciphering key with a server 101 can authenticate that server 101 has secure access to the network static private key SK.network 102b. One benefit of the system depicted in Figure 2a is that the network static private key SK.network 102b does not need to be recorded by or operated with server 101. Further authentication of both parties can be completed via other means including digital signatures in later steps, and the “one-way” authentication in this paragraph refers to the authentication that results from using the ECDH key exchange using at least network static private key SK.network 102b.

Note that the authenticated ECDH key exchange depicted in Figure 2a, with additional details in subsequent Figures, can solve problems in the art discussed in the Description of Related Art. Specifically, through the use of a PK.network 102a recorded by a device and SK.network 102b recorded by a network 105, combined with the use of ephemeral PKI keys for both device 103 and server 101, the depicted and described ECDH key exchange herein can simultaneously achieve both (i) authentication of a network 105 with device 103 and (ii) forward secrecy. As discussed in the Description of Related Art, a device 103 may not have full access to the Internet (such as other servers or networks besides those for a network 105), or other resource limitations such as not storing (x) intermediate certificate authority certificates for servers or (y) compatible parameters or algorithms for intermediate certificate authority certificates for servers, and consequently device 103 may not be able to readily verify a certificate for server 103 such as cert.server 101c without storing and using (x) and (y) above. The mutually authenticated ECDH key exchange with forward secrecy depicted in Figure 2a and subsequent Figures herein supports devices with those limitations. Other benefits are possible as well, such as faster and less resource-intensive authentication of a network 105 with device 103.

After receiving message 206a, server 101 can conduct a point validation step 204a for received value or point X1 211a. Note that point validation step 204a is related to a key validation step 204 and can use several of the same sub-steps depicted and described for a key validation step 204 for server 101 above. A point validation step 204a is different than a key validation step 204 since (i) the

value X1 211a is preferably not used as a public key to be shared with other parties, but rather (ii) represents a point on the ECC curve from parameters 104a that will subsequently undergo a point addition operation in order to mutually derive a shared secret with device 103. Further, point X1 211a can be received through a secure session 221 with a trusted party comprising key server 102, and thus
5 the point X1 211a can have a higher level of confidence or trust as being correct and properly formatted than a device ephemeral public key Ed 103a received potentially via the Public Internet. A point validation step 204a for server 101 can comprise verifying that received point X1 211a is on the ECC curve as specified in parameters 104a and that the point is not the "point at infinity". Other possibilities exist as well for conducting a point validation step 204a on the received point X1 211a without
10 departing from the scope of the present disclosure.

After conducting a point validation step 204a, server 101 can then conduct an ECDH key exchange step 212, where a key exchange step 212 is depicted and described in connection with Figure 2b below. In summary, server 101 can input (i) the server derived ephemeral private key e1 101b from a step 101x and (ii) the received device ephemeral public key Ed 103a from message 203 into an
15 ECDH key exchange algorithm 220 (in Figure 2b) in order to calculate a point X2 212a. Server 101 can then conduct a key derivation step 213 as depicted and described in connection with Figure 2b below. In summary, server 101 can conduct an ECC point addition step 214 (in Figure 2b) using both (i) point X1 211a from message 206b and (ii) point X2 212a from step 212 in order to mutually derive a shared secret X3 213a. Shared secret X3 213a can be input into a key derivation function in order
20 to output a symmetric ciphering key K1 216a and also optionally a MAC key.

Server 101 can then conduct a step 207a to create a digital signature 101s, using an elliptic curve digital signature algorithm (ECDSA) over the values of at least, in part, random number random1 202a and random number random2 205r. The ECDSA could use (i) the private key corresponding to the public key in certificate cert.server 101c as (ii) the private key for creating digital
25 signature 101s in a step 207a. The ECDSA can be compatible with IETF RFC 6979, IETF RFC 4574, and also related FIPS standards or other standards for digital signatures using ECC PKI keys. Additional data to sign for signature 101s in a step 207a could comprise the cryptographic parameters 104a and the certificate cert.server 101c. In addition, other digital signature algorithms besides ECDSA could be used in a step 207a such as the use of RSA based digital signature algorithms, or
30 even post-quantum cryptography algorithms. If other digital signature algorithms besides ECDSA are used in a step 207a, then the public key in certificate cert.server 101c and corresponding private key can support the other digital signature algorithms. In general, the digital signature algorithms used to create digital signature 101s can support cryptographic algorithms and PKI keys that are different than the set of cryptographic algorithms 104 in order to conduct a mutually authenticated ECDH key
35 exchange with forward secrecy as contemplated herein.

Server 101 can then conduct an encryption step 217 (i) using the key K1 216a output from key derivation step 213 in order to (ii) create a ciphertext1 217b. Exemplary details for an encryption step 217 is depicted and described in connection with Figure 2c below, and an encryption step 217 can use a symmetric ciphering algorithm. The plaintext within ciphertext1 217b can comprise at least, in part, the random number random1 202a and random number random2 205r. Other data could be included in plaintext for ciphertext 217b such as the certificate cert.server 101c, digital signature 101s, as well as parameters 104a, without departing from the scope of the present disclosure. For some exemplary embodiments the use or inclusion of a certificate cert.server 101c and digital signature 101s for plaintext in ciphertext 217b could be omitted, since the mutually derived symmetric ciphering key K1 216a can be derived with authentication of server 101 and network 105 to device 103.

Server 101 can then send device 103 a message 206c, where the destination IP address and port number of message 206c can comprise the source IP address and port number 203a received with message 203 and recorded in server database 101d. Message 206c can include the server ephemeral public key E1 101a and the ciphertext1 217b, as depicted in Figure 2a. The value “K1 216a” depicted in Figure 2a is shown to illustrate that the derived symmetric ciphering key 216a from a key derivation step 213 is used to encrypt ciphertext1 217b (indicated by the brackets shown in Figure 2a for message 206c), and the value K1 216a is not normally transmitted as plaintext or ciphertext in message 206c. Ciphertext1 217b can include plaintext values of random number random1 202a, parameters 104a, certificate cert.server 101c, random number random2 205r, and signature 101s. Other data could be included as plaintext in ciphertext 217b such as extensions for a TLS or DTLS handshake, data supporting an application for device 103, and other possibilities exist as well. As depicted in Figure 2a, the series of steps and messages beginning with step 201a for device 103 through the receipt of message 206c by device 103 can comprise a step 222, where the combined step 222 can be used in additional embodiments depicted below.

As contemplated in the present disclosure, a message such as message 206c and also other messages such as message 203, message 206a, etc. can be transmitted or sent in parts, where the data for the message can be transmitted and received in separate datagrams or portions over time. For these embodiments, the message can comprise the collection of separate datagrams or portions transmitted or sent separately. For example, with separate datagrams or portions for a message 206c in Figure 2a, a first datagram or portion for message 206c could comprise server ephemeral public key E1 101a, which could be sent (i) after a key pair generation step 101x, and (ii) before receiving message 206a from key server 102. A second datagram or portion for message 206c could comprise ciphertext1 217b, which could be sent after server 101 receives message 206a from key server 102. In this manner, by sending message 206c as a first portion and a second portion, the overall speed of conducting a step 223 for device 103 could be increased. For example, by receiving the first portion of message 206c comprising key E1 101a, device 103 could then (a) begin conducting steps below of 204 and 218,

while (b) waiting for the second portion of message 206c comprising ciphertext1 217b to be sent separately and after the first portion. By increasing the overall speed for conducting a step 223 for device 103, then electrical power consumption or battery usage for device 103 can be reduced. Other possibilities and benefits from sending a message in the present disclosure as a first portion and a second portion, without departing from the scope of the present disclosure. Messages depicted and described herein may be sent and received as multiple portions over time, where the message can comprise the collection of the multiple portions.

Device 103 can then receive message 206c and conduct a series of steps in order to process the message. Device 103 can conduct a key validation step 204 in order to verify that server ephemeral public key E1 101a in message 206c is properly formatted and is a valid point on the named curve for parameters 104a. Validation step 204 for device 103 can be equivalent to the validation step 204 for server 101 described above. Device 103 can then conduct an ephemeral ECDH (ECDHE) key exchange step 218 in order to mutually derive symmetric ciphering key K1 216a. Details for an ECDHE key exchange step 218 is depicted and described in connection with Figure 2c below. In summary, device 103, using parameters 104a, can perform an elliptic curve point addition operation on (i) the server ephemeral public key E1 101a received in message 206c and (ii) the recorded network static public key PK.network 102a. Device 103 can input (i) the point derived from ECC point addition and (ii) the device ephemeral private key ed 103b into an ECDH key exchange algorithm in order to mutually derive shared secret key X3 215 with server 101. The mutual derivation of shared secret key X3 215 by server 101 is depicted and described in connection with key exchange step 213 for server 101 in Figure 2b below. Device 103 can input shared secret key X3 215 into a key derivation function in order to mutually derive symmetric ciphering key K1 216a. Note that a MAC key could also be derived in step 218.

Device 103 can then perform a decryption step 219 in order to decrypt ciphertext1 217b from message 206c using the derived symmetric ciphering key K1 216a from the key exchange step 218, where symmetric ciphering key K1 216a was derived as described in the paragraph above. A decryption step 219 is also depicted and described in connection with Figure 2c below. Device 103 can then read the plaintext within ciphertext1 217b, as well as verifying message integrity of ciphertext1 217b using a MAC key derived in a step 218. Device 103 in a decryption step 219 can read the plaintext values of random number random1 202a, random number random2 205r, and certificate cert.server 101c, as well as a digital signature 101s. Note that digital signature 101s can be over at least the random number random1 202a that device 103 sent in a message 203.

At step 208, device 103 can conduct a verification step for the plaintext certificate cert.server 101c in order to validate the certificate. Device 103 in a step 208 can verify a signature from a certificate authority for the server static public key PK.server 101p in the certificate (plus any intermediate certificate signatures) using a root certificate for the certificate authority. The root

certificate for the certificate authority could be recorded in a nonvolatile memory for device 103. Device 103 can verify both the certificate authority signature in cert.server 101c using an elliptic curve digital signature algorithm (ECDSA). The ECDSA could use a certificate authority public key for from a root certificate for verifying the certificate authority signature in a certificate cert.server 101c.

5 The ECDSA can be compatible with IETF RFC 6979, IETF RFC 4574, and also FIPS 186-4 standards or related and subsequent standards for digital signatures using ECC PKI keys.

Note that a certificate cert.server 101c could also specify parameters different than the use of an ECC algorithm, such as using RSA based signatures. For these embodiments using RSA based keys for digital signatures, device 103 could use a digital signature algorithm (DSA) and server static

10 public key PK.server 101p can comprise an RSA-based key. Note that in some exemplary embodiments, the use of a server certificate cert.server 101c could be omitted, since device 103 can authenticate server 101 using the authenticated ECDH key exchange step 218 (where successful decryption of ciphertext1 217b proves to device 103 that server 101 has access to SK.network 102b). Further, a server certificate cert.server 101c could be included in a message 206c and ciphertext1 217b,

15 but device 103 could omit a separate certificate verification step 208 and still trust the server public key PK.S1 101p in a cert.server 101c. In other words, successful decryption of the cert.server 101c with the symmetric ciphering key K1 216a can signal or indicate that cert.server 101c can be trusted using the stored PK.network 102a, since the cert.server 101c could only be encrypted by a server 101 with access to SK.network 102b.

20 After a step 208 to verify certificate cert.server 101c, device 103 can conduct a signature verification step 209a to verify signature 101s. For a step 209, device 103 could use the server static public key PK.server 101p for server 101 from certificate cert.server 101c and an ECDSA signature algorithm in order to verify signature 101s. The signed data verified by a signature verification step 209a can comprise at least, in part, both random number random1 202a from device 103 and random

25 number random2 205r from server 101, as well as other data within message 206c such as certificate cert.server 101c. If the signature verification step 209a fails, then device 103 can stop further processing of message 206c and return an error message.

Device 103 can conduct a signature creation step 207b in order to create digital signature 103s over data received in message 206c. The data signed by a signature creation step 207b for signature

30 103s can comprise at least, in part, random number random2 205r. A set of parameters 104a can specify values and settings to utilize with an ECDSA in a step 209a, such as a secure hash algorithm to utilize, the use of a deterministic ECC signature algorithm (avoiding the need to include a unique random number from device 103 with the signature 103s), padding rules, encoding rules, etc. Device 103 can use device private key d1 101d in order to create signature 103s.

35 Device 103 can then conduct an encryption step 217c, where encryption step 217c can use the exemplary encryption step 217 depicted and described below in Figure 2c with different plaintext data

than the depicted data for a step 217 in Figure 2c. The encryption key for a step 217c can comprise the symmetric ciphering key K1 216a derived by device 103 above in a step 218, and a MAC key 216b (from Figure 2c below) can also be utilized. In some exemplary embodiments, the encryption step 217c can use a different symmetric ciphering key K1 216a than key K1 216a used by server 101 to encrypt ciphertext1 217b. In other words, different symmetric ciphering keys could be used by (i) server 101 to encrypt ciphertext1 217b and (ii) device 103 to encrypt a ciphertext 217d. However, both server 101 and device 103 can mutually derive the different symmetric ciphering keys using at least the mutually derived shared secret X3 215. For some exemplary embodiments, the key K1 216a from a KDF 216 can comprise two portions, where (i) a first portion is used by server 101 to encrypt data and device 103 to decrypt data and (ii) a second portion is used by device 103 to encrypt data and server 101 to decrypt data.

The plaintext data for an encryption step 217c can comprise at least, in part, an identity for device 103 of ID.device 103i, and the random number random2 205r from server 101. Other data could be included in the plaintext for an encryption step 217c without departing from the scope of the present disclosure, such as, but not limited to, data from a transducer connected to device 103. In addition, the device 103 static public key D1 103c, or a certificate for device 103 with public key D1 103c could be included as plaintext data for an encryption step 217c. The output of an encryption step 217c can comprise ciphertext2 217d, as depicted in Figure 2a. As depicted and described in connection with Figure 2c below, the output of an encryption step 217c could also include an initialization vector and a MAC code, which could be included as metadata or plaintext along with ciphertext2 217d in a message 210a. The initialization vector can be used to chain blocks in order to scramble data across the multiple blocks and the MAC code can be used to confirm message integrity using a MAC key output from key exchange algorithm 218. For embodiments where server 101 could store or receive device static public key D1 103c before receiving a message 210a (such as receiving the key D1 103c from a server associated with device 103), then key D1 103c and/or a certificate for device 103 could be omitted from ciphertext2 217d and a message 210a.

After step 217c, device 103 can send server 101 a message 210a, where message 210a can include ciphertext2 217c. In exemplary embodiments, message 210a is transmitted by device 103 using the same source IP address and port number as message 203. In addition, message 210a is transmitted by device 103 using the same destination IP address and port number for server 101 as message 203. Although the signature 103s is depicted in Figure 2a as being internal to ciphertext2 217c, in some exemplary embodiments signature 103s can be external to ciphertext2 217c. Likewise, although a signature 101s is depicted as within a ciphertext 217b from server 101, in some embodiments a signature 101s could be external to ciphertext 217b in a message 206c. Server 101 can receive message 210a by listening to the same local IP address and port number used to receive message 203 above.

After server 101 receives message 210a, server 101 can conduct a series of steps in order to process the message. Server 101 can conduct a decryption step 219a, which can comprise a decryption step 219 depicted and described below in connection with Figure 2c, but with different ciphertext data. The ciphertext data for a decryption step 219a can comprise the ciphertext2 217c received by server 101 in message 210a. A decryption step 219a can also use an initialization vector and MAC code received along with ciphertext2 217c in message 210a. After conducting a decryption step 219a, server 101 can read the plaintext data within ciphertext2 217c. In exemplary embodiments, the plaintext data can include an identity for device 103 of ID.device 103i, the device static public key D1 103c, and also the random number random2 205r. Although not depicted in Figure 2a, ciphertext2 217a as received by server 101 can include input from a transducer or sensor operated by device 103, such as, but not limited to, keyboard input, temperature data from a thermocouple or thermistor, pressure data from a transducer, the state of an actuator, the state of an electronic switch, gate, or relay, etc. operated by device 103. Other possibilities exist as well for transducer data in ciphertext2 217a which is decrypted into plaintext by server 101 in a decryption step 219a without departing from the scope of the present disclosure.

At step 210b, server 101 can process the plaintext data output from a decryption step 219a. Server 101 can read and record the device identity ID.device 103i for use in subsequent messages. Server 101 can read the value for random number random2 205r to confirm the value or number equals the random number random2 205r sent above in message 206c. In exemplary embodiments, server 101 can record the plaintext data decrypted from ciphertext2 217c in a server database 101d along with a timestamp, after completing the signature verification step 209c. Server 101 can conduct a signature verification step 209b for signature 103s using the same signature verification algorithm and parameters as signature verification step 209a, except using the device static public key D1 103c. Parameters 104 can specify settings or values for conducting a signature verification step 209a. In exemplary embodiments, signature verification step 209b comprises an ECDSA signature verification for digital signature 103s using key D1 103c. Note that signature 103s is over data that includes at least random number random2 205r sent by server 101 in message 206c. Device static public key D1 103c could be recorded in nonvolatile memory or disk storage of server 101 as depicted in Figure 1b above.

Upon successful completion of a signature verification step 209b for digital signature 103s, server 101 and device 103 can conduct additional steps to securely transfer data 106 between the two nodes. Although not depicted in Figure 2a, server 101 could send device 103 commands, files, configuration data, or other data using ciphertext encrypted with derived symmetric ciphering keys. Server 101 and device 103 could also update key K1 216a or rotate key K1 216a using a key derivation function (such as key derivation function 216 depicted in Figure 2b and Figure 2c below). As depicted in Figure 2a, after a step 210b and a step 209b, server 101 can send key server 102 a message 210b,

where message 210b can include the device identity ID.device 103i and an “OK” message, where the “OK” signals to key server 102 that server 101 and device 103 have successfully derived and used symmetric ciphering key 216a using PKI keys and an ECDH point addition of shared secret X1 211a and X2 212a. As depicted in Figure 2a, the series of steps beginning with a step 204 for device 103 through the receipt of message 210b can collectively comprise a step 223.

Figure 2b

Figure 2b is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive shared secrets, and for conducting a key derivation function using the derived shared secrets, in accordance with exemplary embodiments. Key server 102 can conduct a key exchange step 211 in order to derived a secret key X1 211a. Server 101 can conduct a key exchange step 212 in order to derive a secret key X2 212a. Server 101 can receive the secret key X1 211a in a message 206b from key server 102 in Figure 2a above through a secure connection 221. Server 101 can then conduct a key derivation function 213 using shared secrets X1 211a and X2 212a in order to derive a symmetric ciphering key K1 216a. Using the methods and ECC PKI keys described in the present disclosure, a device 103 can also derive the same symmetric ciphering key K1 216a as depicted and described below for a key exchange step 218 in Figure 2c. In other words, for exemplary embodiments (i) the corresponding key exchange step 218 (in Figure 2c below) for a device 103 by network 105 can be (ii) shared or distributed between a server 101 and key server 102 in order to secure or isolate network static private key SK.network 102b.

The processes and operations, described below with respect to all of the logic flow diagrams and flow charts may include the manipulation of signals by a processor and the maintenance of these signals within data structures resident in one or more memory storage devices. For the purposes of this discussion, a process can be generally conceived to be a sequence of computer-executed steps leading to a desired result.

These steps usually require physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is convention for those skilled in the art to refer to representations of these signals as bits, bytes, words, information, elements, symbols, characters, numbers, points, data, entries, objects, images, files, or the like. It should be kept in mind, however, that these and similar terms are associated with appropriate physical quantities for computer operations, and that these terms are merely conventional labels applied to physical quantities that exist within and during operation of the computer.

It should also be understood that manipulations within the computer are often referred to in terms such as listing, creating, adding, calculating, comparing, moving, receiving, determining,

configuring, identifying, populating, loading, performing, executing, storing etc. that are often associated with manual operations performed by a human operator. The operations described herein can be machine operations performed in conjunction with various input provided by a human operator or user that interacts with the device, wherein one function of the device can be a computer.

5 In addition, it should be understood that the programs, processes, methods, etc. described herein are not related or limited to any particular computer or apparatus. Rather, various types of general purpose machines may be used with the following process in accordance with the teachings described herein.

10 The present invention may comprise a computer program or hardware or a combination thereof which embodies the functions described herein and illustrated in the appended flow charts. However, it should be apparent that there could be many different ways of implementing the invention in computer programming or hardware design, and the invention should not be construed as limited to any one set of computer program instructions.

15 Further, a skilled programmer would be able to write such a computer program or identify the appropriate hardware circuits to implement the disclosed invention without difficulty based on the flow charts and associated description in the application text, for example. Therefore, disclosure of a particular set of program code instructions or detailed hardware devices is not considered necessary for an adequate understanding of how to make and use the invention. The inventive functionality of the claimed computer implemented processes will be explained in more detail in the following description in conjunction with the remaining Figures illustrating other process flows.

20 Further, certain steps in the processes or process flow described in all of the logic flow diagrams below must naturally precede others for the present invention to function as described. However, the present invention is not limited to the order of the steps described if such order or sequence does not alter the functionality of the present invention. That is, it is recognized that some steps may be performed before, after, or in parallel other steps without departing from the scope and spirit of the present invention.

25 The processes, operations, and steps performed by the hardware and software described in this document usually include the manipulation of signals by a CPU or remote server and the maintenance of these signals within data structures resident in one or more of the local or remote memory storage devices. Such data structures impose a physical organization upon the collection of data stored within a memory storage device and represent specific electrical or magnetic elements. These symbolic representations are the means used by those skilled in the art of computer programming and computer construction to most effectively convey teachings and discoveries to others skilled in the art.

30 A key exchange step 211 for key server 102 to derive a secret key X1 211a can utilize a selected set of cryptographic parameters 104a as depicted and described in connection with Figure 1a and Figure 2a above. As depicted in Figure 2b, an key exchange algorithm 220 in step 211 for key

server 102 can receive input both of device ephemeral public key Ed 103a and network static private key SK.network 102b. The key exchange algorithm 220 could comprise a Diffie Hellman key exchange (DH), an Elliptic Curve Diffie Hellman key exchange (ECDH), and other possibilities exist as well without departing from the scope of the present invention. A key exchange algorithm 220 can support either PKI keys based on elliptic curves or RSA algorithms, although support of elliptic curves may be preferred in some exemplary embodiments due to their shorter key lengths and lower computational processing requirements.

A summary of ECDH as a key exchange algorithm 220 is included in the Wikipedia article titled "Elliptic Curve Diffie-Hellman" from March 9, 2018, which is herein incorporated by reference. An exemplary embodiment of key exchange algorithm 220 could comprise a "One-Pass Diffie-Hellman, C(1, 1, ECC CDH)" algorithm as described in section 6.2.2.2 on page 81 of the National Institute of Standards and Technology (NIST) document "NIST SP 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" from March, 2007 which is hereby incorporated by reference its entirety. Other key exchange algorithms in NIST SP 800-56A could be utilized as well for a key exchange algorithm 220 in Figure 2a and Figure 2b without departing from the scope of the present disclosure. Example calculations for an ECDH key exchange for a key exchange algorithm 220 are shown below in Figure 2c.

Other algorithms to derive a secret keys using public keys and private keys may also be utilized in a key exchange algorithm 220, such as, but not limited to, the American National Standards Institute (ANSI) standard X-9.63. Cryptographic parameters 104a can also include information, values, or settings for conducting (i) a key exchange algorithm 220 in step 211 and step 212 and (ii) a key derivation function 216 in order to derive a commonly shared symmetric encryption key K1 216a. As contemplated herein, the terms "selected set of cryptographic parameters 104a" and "cryptographic parameters 104a", and "parameters 104a" can be equivalent, and can also comprise a subset of exemplary cryptographic parameters depicted and described in connection with Figure 1a and Figure 2d below. Parameters 104a input into a key exchange algorithm 220 can include a time-to-live for a key K1 216a that is derived, a supported point formats extension, where the supported point formats extension could comprise uncompressed, compressed prime, or "compressed char2" formats, as specified in ANSI X-9.62. In other words, (i) an ECC keys input into a key exchange algorithm 220 and (ii) secret keys output from key exchange algorithm 220 may have several different formats and a set of parameters 104a can be useful to specify the format. As depicted in Figure 2b, the output of a key exchange algorithm 220 in a step 211, such as an ECDH key exchange, can comprise a secret value X1 211a. In exemplary embodiments, secret value X1 211a can comprise a point on an elliptic curve, where the equation and values for the elliptic curve can be specified in parameters 104a. As contemplated herein, the secret value X1 211a (as well as X2 212a below) comprises both an X coordinate and a Y coordinate, in order to support subsequent ECC point addition operations.

Key exchange step 212 for a sever 101 depicted in Figure 2a can correspond to key exchange 212 in Figure 2b. Key exchange step 212 can comprise inputting or using the device ephemeral public key Ed 103a (from message 203 in Figure 2a) and the server ephemeral private key e1 101b (from a key generation step 101x) into a key exchange algorithm 220, which can comprise the same or equivalent key exchange algorithm 220 depicted and described in connection with key exchange step 211 described above. Other elements or algorithms within a key exchange step 212 can be equivalent to a key exchange step 211 above, including the use of shared parameters 104a. The output of a key exchange algorithm 220 in a step 212 can comprise a secret key or value X2 212a. In exemplary embodiments, secret value X2 212a can comprise a point on an elliptic curve, where the equation and values for the elliptic curve can be specified in parameters 104a. Exemplary numeric values for using a key exchange algorithm 220 are depicted and described below, and key exchange algorithm 220 can utilize an ECC point multiplication of a public key by the scalar value of a private key. In exemplary embodiments, a server 101 can record the value X2 212a derived from a step 212 and also the value X1 211a received in a message 206b in a server database 101d. The time the values are stored in a server database 101d can be minimized in order to increase security, and, for example, the recording of the values can be deleted before server 101 sends the "OK" message 210b to key server 102 in Figure 2a.

A key derivation step 213 for server 101 can (i) combine the output of key exchange steps 211 and 212 in order to calculate or derived the shared secret X3 215 and then (ii) perform a key derivation function step 216 on the derived or calculated shared secret X3 215 in order to determine or calculate shared secret key K1 216a, which can comprise a symmetric ciphering key. Note that shared secret key K1 216a can be also mutually derived by device 103, where device 103 uses the key exchange step 218 depicted and described in connection with Figure 2c below. In exemplary embodiments, a server 101 can conduct the key derivation step 213 using (i) the value X1 211a received from key server 102 (where receipt of X1 211a by server 101 can be in a message 206b as shown in Figure 2a above), and (ii) the value or key X2 212a output from a key exchange step 212 for server 101 in the paragraph above. As contemplated herein, the values of X1 211a, X2 212a, and X3 215 may be described as either "shared secrets" or "shared secret keys". Although the values may not normally be used as a key directly with a symmetric ciphering algorithm, these values and the output of a key exchange algorithm 220 can comprise a secret or a key.

Key derivation step 213 for server 101 can comprise two primary steps. A first step in key derivation 213 can comprise an ECC point addition 214 on the value X1 211a and the value X2 212a. The result of the ECC point addition will be equal to the value X3 215. Note that device 103 can also derive the same value for value X3 215 (in step 218 below) without ECC point addition 214 using a step 218. In other words, although (a) the related key exchange step 218 for device 103 may include a point addition for public keys, (b) the key exchange step 218 for device 103 will not use ECC point

addition for points derived from two separate private keys in two separate servers (e.g. X1 211a uses private key SK.network 102b and X2 212a uses private key e1 101b).

Exemplary calculations for an ECC point addition 214 can comprise the calculations shown for point addition in the Wikipedia article for “Elliptic Curve Point Multiplication” dated May 15, 2018, which is herein incorporated by reference in its entirety. As depicted in Figure 2b, (a) the calculation of X3 215 by server 101 using an ECC point addition 214 over X1 211a and X2 212a will equal (b) the value for X3 215 calculated by device 103 using a key exchange algorithm 220 in a step 218 from Figure 2c below. A second step in key derivation step 213 as depicted in Figure 2b can comprise a key derivation function step 216 using (a) input from ECC point addition step 214 (e.g. value X3 215 output from step 214), where (b) the output of key derivation function step 216 can comprise key K1 216a and also an associated MAC key 216b. In exemplary embodiments, the X coordinate from shared secret X3 215 can be used with key derivation function 216.

By server 101 conducting a key derivation step 213 as depicted in Figure 2b (where key server 102 conducts the calculations for step 211 using the network static private key SK.network 102b), (i) sever 101 can calculate symmetric ciphering key K1 216a without recording or operating on the network static private key SK.network 102b. In this manner, the security of a system 100 or system 200 can be significantly enhanced, since the network static private key 102b does not need to be recorded or operated by server 101, which can communicate with a plurality of devices 103 over an IP network. In other words, by server 101 (i) using the ECC point addition over key X1 211a instead of (ii) conducting a key exchange 220 directly with SK.network 102b, then server 101 does not need to record or operate with the network static private key SK.network 102b, thereby increasing security. Also, since (i) key X1 211a can be the equivalent of an ECC public key as a point on an elliptic curve, and (ii) it is not computationally feasible to determine network static private key SK.network 102b from key X1 211a, then key X1 211a does not reveal meaningful information about network static private key SK.network 102b.

Many benefits can be achieved by server 101 conducting a key derivation step 213 using key X1 211a instead of recording and operating with network static private key SK.network 102b. As one example, the corresponding network static public key PK.network 102a could potentially be both (i) recorded in millions of distributed devices connecting to server 101 through many different physical locations and networks, and (ii) used for a decade or longer. Keeping network static private key SK.network 102b secure for this embodiment could be economically essential, since a compromise of network static private key SK.network 102b may (i) render the devices 103 insecure (or unable to authenticate network 105 using an ECDHE key exchange), and (ii) require the secure distribution or re-installation of a new, different network static public key SK.network 102a in the devices, which may not be economically feasible due to the prior distribution of devices.

Exemplary data and numbers can be provided to demonstrate the calculations for (i) key exchange step 211, (ii) key exchange step 212, and (iii) key derivation step 213 using an ECC point addition 214. The exemplary data can comprise decimal numbers for the example ECC PKI keys and exchanged keys listed in “Test vectors for DPP Authentication using P-256 for mutual authentication”
 5 on pages 88 and 89 of the DPP specification version 1.0. Parameters 104a can comprise the elliptic curve of “secp256r1” with key lengths of 256 bit long keys.

The network static private key SK.network 102b can comprise the exemplary following number, and can be recorded in key server 102:

10 38358416135251014160802731750427376395128366423455574545250035236739593908128

The server ephemeral private key e1 101b can comprise the exemplary following number, and can be recorded by server 101:

15 111991471310604289774359152687306247761778388605764559848869154712980108827301

The device ephemeral public key Ed 103a can comprise the following exemplary values with X and Y numbers (or “coordinates”) of:

20 X:
 61831688504923817367484272103056848457721601106987911548515219119661140991966
 Y:
 436821274116052626307636850969789027573720854595612820926922498255090826944

25 Key exchange step 211 for an ECDH algorithm key exchange 220 by key server 102 can input the device ephemeral public key Ed 103a and the network static private key SK.network 102b (both with numbers above) in order to calculate a secret X1 211a. An exemplary number or value for secret X1 211a from the values above using parameters 104a can be:

30 X:
 11490047198680522515311590962599671482029417064351337303313906642805743573119
 Y:
 27933966560238204731245097943399084523809481833434754409723604970366082021855

35 Key exchange step 212 for an ECDH algorithm key exchange 220 by server 101 can input the device ephemeral public key Ed 103a and the server ephemeral private key e1 101b (both with

numbers above) in order to calculate a secret X2 212a. An exemplary number or value for key X2 212a from the values above using parameters 104a can be:

X:

5 78944719651774206698250588701582570633503182903415394243006529481189158194650

Y:

11227712702924684581834935828837489140201820424536062912051086382324589445237

10 An ECC point addition 213 for the above two derived points (or “keys”) X1 211a (from keys Ed 103a and SK.network 102b) and X2 212a (from keys Ed 103a and e1 101b) will result in the following point that also equals X3 215.

X:

113734500629065545557893524064610113740858966831672649615565042035695230713090

15 Y:

68961429691307429166796760881095689348088875771334970644593306388375741965262

20 Note that the same numeric value for key X3 215 can also be derived by device 103 from a key exchange step 218 below using ECDH key exchange algorithm 220a. For exemplary embodiments, although private key SK.network 102b and ephemeral private key e1 101b are recorded and operated by physically separated devices, device 101 can record and operate on the corresponding public keys PK.network 102a and ephemeral public key E1 101a (at the same physical location as device 103).

25 After an ECC point addition 213, for a key derivation step 218 in Figure 2b, server 101 can input the shared secret key X3 215, where key X3 215 was output from the ECC point addition 214, into a key derivation function 216. The key derivation function 216 can comprise the same key derivation function 216 used by a device 103 in a step 218 below. The output of a key derivation function 216 can comprise both (i) a symmetric ciphering key K1 216a and (ii) a MAC key 216b. MAC key 216b can be used with a symmetric ciphering algorithm in order to generate a MAC code, 30 such that the other party using the same key K1 216a and MAC key 216b can process the ciphertext and calculate the same MAC code in order to verify message integrity.

35 Key derivation function 216 can use a secure hash function such as, but not limited to, SHA-256, SHA-384, SHA-3, etc. and additional values such as a text string with secret X3 215. The specification of a secure hash algorithm and the text string for use with a key derivation function 216 could be commonly shared between server 101 and device 103 by commonly shared parameters 104a. In some exemplary embodiments, the text string for use with secret X3 215 can be from data, text, or

values transmitted in (i) message 203 (for KDF 216 by server 101 in step 213) and/or (ii) message 206c (for KDF 216 by device 103 in step 218). The output of a secure hash algorithm within a key derivation function 216 could have a subset of bits selected or possibly a secure hash expanded in order to obtain the number of bits required for a symmetric key with a symmetric ciphering algorithm, such as key K1 216a. A key derivation function (KDF) 216 could comprise a KDF compatible with or specified by ANSI standards for "X9.63 Key Derivation Function". Other possibilities exist for a key derivation function 216 to convert a secret X3 215 into a symmetric ciphering key K1 216a and a MAC key 216b without departing from the scope of the present disclosure. As contemplated in the present disclosure, although an ECC public key such as secret X3 215 can comprise a coordinate with an X value and a Y value, in exemplary embodiments a single number comprising the X value can be selected and input into a key derivation function 216. In addition, the key K1 216a can comprise two portions, where (i) a first portion can be a key for encrypting data by server 101 and decrypting the data by device 103 and (ii) a second portion can be a key for encrypting data by device 102 and decrypting the data by server 101.

Figure 2c

Figure 2c is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key, and for using the derived shared secret key to encrypt and decrypt data, in accordance with exemplary embodiments. Exemplary steps for a device 103 to mutually derive a shared secret X3 215 and symmetric key 216a can comprise a key exchange step 218. Exemplary steps in Figure 2c for a server 101 to encrypt plaintext data using the mutually derived symmetric key 216a can comprise an encryption step 217. Exemplary steps in Figure 2c for a device 103 to decrypt ciphertext data using the mutually derived symmetric key 216a can comprise a decryption step 219. The use of the steps for a key exchange 218, encryption 217, and decryption 219 were also depicted and described in connection with Figure 2a above. Note that steps in Figure 2c and the steps in Figure 2b can share some algorithms and values, and the descriptions for the algorithms and values in Figure 2ab can be applicable for Figure 2c. For example, the key exchange algorithm 220a can comprise an ECDH key exchange equivalent to key exchange step 220. The set of parameters 104a depicted and described in Figure 2b can also be used in Figure 2c.

A device 103 can conduct a key exchange step 218. At step 218, (i) a combination of a recorded network static public key PK.network 102a and received server ephemeral public key E1 101a, and (ii) the derived device ephemeral private key ed 103b can be input into an ECDH key exchange algorithm 220a in order to calculate the shared secret X3 215. The recorded network static public key PK.network 102a and received server ephemeral public key E1 101a can be combined via elliptic curve point addition. Exemplary data and numbers can be provided to demonstrate the

calculations for (i) key exchange step 218. The exemplary data can comprise decimal numbers for the example ECC PKI keys and exchanged keys listed in “Test vectors for DPP Authentication using P-256 for mutual authentication” on pages 88 and 89 of the DPP specification version 1.0. Parameters 104a can comprise the elliptic of “secp256r1” with key lengths of 256 bit long keys.

5

The device ephemeral private key ed 103b can comprise the exemplary following number, and can be recorded in device 103 after a key pair generation step 103x from Figure 1a above:

9814229718244518553550958692061024829480281279450793086167684747145642004923

10

The network static public key PK.network 102a can comprise the exemplary values with X and Y numbers (or “coordinates”) of:

X:

4419807000381358656111506147651622980270029110554119329493335953912822452287

15

Y:

37427159939572325965354914097696269740713866333885143374269952770772578794844

The server ephemeral public key E1 101a can comprise the following exemplary values with X and Y numbers (or “coordinates”) of:

20

X:

42629956901026513598149966301519681371972968598637962756879877886841583606416

Y:

20486612594265388212565154850034967164732043090221075006612427172869133074917

25

An ECC point addition for the above two keys E1 101a and PK.network 102a will result in the following exemplary point. which comprises (a) both E1 101a and PK.network 102a for a key exchange step 218 then (b) input into an ECDH key exchange algorithm 220a:

X:

2811461365732647553134637541685353169648905058941523144737599092152119800180

30

Y:

93903335977032690345879985966890561591048675256101157964834025539587687968435

The above combination of both E1 101a and PK.network 102a for a key exchange step 218 via an ECC point addition operation is depicted in Figure 2c with the “+” symbol between the public keys.

35

The output of the above ECC point addition for public keys E1 101a and PK.network 102a can be input into ECDH key exchange algorithm 220a using parameters 104a. All of the exemplary calculations for a key exchange step 218 can use the exemplary subset of cryptographic parameters 104a. An ECDH algorithm key exchange 220a in key exchange step 218 can input (i) the exemplary point immediately above from the ECC point addition operation on the public keys 101a and 102a and (ii) the device ephemeral private key ed 103b into the ECDH key exchange 220a, and output the point X3 215. Note that the secret X3 215 as derived by device 103 in a key exchange step 218 equals or is the same numeric value as the secret X3 215 derived by server 101 in a key derivation step 213 in Figure 2b. An exemplary number or value for secret X3 215 calculated by device 103 using a key exchange step 218 using the above exemplary numeric values for ed 103b, PK.network 102a, and E1 101a would be:

X:

113734500629065545557893524064610113740858966831672649615565042035695230713090

Y:

68961429691307429166796760881095689348088875771334970644593306388375741965262

Although Figure 2c depicts an ECC point addition operation over public keys E1 101a and PK.network 102a, the same shared secret value X3 215 could be generated or derived by conducting (i) a first ECC point multiplication operation with the server ephemeral public key E1 101a and the device ephemeral private key ed 103b to derive a first point, and (ii) a second ECC point multiplication operation with the network ephemeral public key PK.network 102a and the device ephemeral private key ed 103b to derive a second point, and (iii) an ECC point addition operation with the first point and the second point to derive the shared secret value X3 215. In other words, the value X3 215 can be calculated as either:

$$(i) \quad X3\ 215 = [E1\ 101a + PK.network\ 102a] * ed\ 103b, \text{ or}$$

$$(ii) \quad X3\ 215 = [E1\ 101a * ed\ 103b] + [PK.network\ 102a * ed\ 103b]$$

For a key derivation step 218, derived shared secret key X3 215 can be input into a key derivation function 216 where the key derivation function 216 can be equivalent to the key derivation function 216 depicted and described in connection with Figure 2b above for a key derivation step 213. Note that for key derivation steps in the present disclosure, the X coordinate of a derived shared secret can be taken or used as input into the key derivation function. The output of a key derivation function 216 can comprise both (i) a symmetric ciphering key K1 216a and (ii) a MAC key 216b. MAC key 216b can be used with a symmetric ciphering algorithm in order to generate a MAC code, such that the other party using the same key K1 216a and MAC key 216b can process the ciphertext and

calculate the same MAC code in order to verify message integrity. The use of key K1 216a and MAC key 216b are described in connection with encryption step 217 and decryption step 219.

Server 101 can conduct an encryption step 217, where the use for an encryption step 217 is depicted and described in connection with Figure 2a above. Plaintext 217a in a step 217 can comprise the first random number random1 202a from device 103, the second random number random2 205r, and the server certificate cert.server 101c. Other or different exemplary data could be included as plaintext 217a in an encryption step 217, including extensions for a TLS or DLTs handshake. The symmetric ciphering key for encryption step 217 can comprise symmetric key K1 216a from a key derivation step 213 and a MAC key 216b can be input into a symmetric ciphering algorithm 225 as well. Encryption step 217 and decryption step 219 can use a common symmetric ciphering algorithm 225, which could comprise the Advanced Encryption Standard with Synthetic Initialization Vectors (AES-SIV) (and deciphering algorithm) also with a common set of symmetric ciphering parameters 104f from a set of cryptographic parameters 104. Other or different symmetric ciphering algorithms 225 could be utilized as well, such as, but not limited to such as AES, Triple Data Encryption Standard (3DES), Blowfish, or related algorithms. A mutually derived symmetric ciphering key K1 216a can comprise two portions, where a first portion is used by server 101 for encryption and a second portion is used by device 103 for encryption. At least the first portion of key K1 216a can be used in an encryption step 217.

Symmetric ciphering parameters 104f can also specify the use of a block chaining mode such as cipher block chaining (CBC), counter mode (CTR), or Galois/Counter mode (GCM) and other possibilities exist as well. In addition, symmetric ciphering parameters 104f could specify a mode for message authentication, which could comprise a CMAC mode as specified in NIST publication SP-800-38B. In some exemplary embodiments, a symmetric ciphering algorithm 225 can comprise the AES-SIV algorithm as specified in IETF RFC 5297. The output from an encryption step 217 using a symmetric ciphering algorithm 225 and the depicted values input can be ciphertext 217b, as depicted in Figure 2c.

A decryption step 219 can be performed by device 103. A decryption 219 step converts the ciphertext 217b received in a message 206c from Figure 2a into plaintext 217a. Decryption step 219 can utilize a symmetric decryption algorithm 225, which could comprise the same algorithm used in symmetric encryption algorithm 225 except the algorithm being used for decryption instead of encryption. Note that the same values are input into symmetric decryption algorithm 225 as symmetric encryption algorithm 225 above, such as symmetric encryption key K1 216a (or the first portion of key K1 216a if a second portion of key K1 216a is used by device 103 for encryption) and parameters 104f in order to convert ciphertext 217b back into plaintext 217a. Additional data input into symmetric decryption algorithm 211b can comprise an initialization vector 217i and MAC code 216c which could be sent along with ciphertext 217b.

Device 103 can read and process plaintext 217a after a decryption 219 step. The plaintext 217a as read by device 103 can comprise the first random number random1 202a from device 103, the second random number random2 205r, and the server certificate cert.server 101c. In exemplary embodiments, the successful decryption of a ciphertext into a plaintext using decryption algorithm 225 supports one-way authentication of the server 101 and/or network 105, since successful decryption by device 103 can only take place when the server 101 has access to network static private key SK.network 102b. In other words, only the nodes could mutually derive key K1 216a in Figure 2b and Figure 2c by (i) device 103 recording PK.network 102a and (ii) server 101 having access to SK.network 102b (via key server 102). Thus, data that is successfully encrypted by the server 101 and decrypted by the device 103 using key K1 216a would confirm the server 101 is authenticated.

As depicted and described in connection with Figure 2a, server 101 or device 103 can also conduct both an encryption step 217 and a decryption step 219. The steps for server 101 to conduct a decryption step 219 for can comprise step 219a as depicted and described in Figure 2a. When server 101 conducts decryption step 219a using symmetric encryption key K1 216a, the ciphertext and plaintext will comprise different values than those depicted in Figure 2c, where the ciphertext for a decryption step 219a can comprise ciphertext2 217d. Further, a device 103 can conduct an encryption step 217c in with key K1 216a in order to create ciphertext 217d, as depicted in Figure 2a.

Figure 2d

Figure 2d is an illustration of an exemplary server database and an exemplary set of cryptographic parameters, in accordance with exemplary embodiments. A server database 101d depicted and described above in connection with system 100 and system 200 can record data for server 101 to work with a plurality of devices 103 and at least one key server 102. A server database 101d could record in at least one set of values, keys, and/or numbers for a plurality of devices 103. Other possibilities exist as well for the organization, tables, and recorded data within a server database 101d without departing from the scope of the present disclosure. Data within server database 101d could be encrypted using a symmetric key. Although system 100 and system 200 depict a server database 101d as operating or recorded within a server 101, a server database 101d could comprise a separate server within a network 105 and communicating with server 101 via a secure session 221 or a private network 107a. Further, a server database 101d, when operating or recorded in a separate server than server 101, then server database 101d could contain electrical components equivalent to a server 101 depicted and described in connection with Figure 1b.

Server database 101d can record values or numbers for a first random number random1 202a, received device ephemeral public key Ed 103a, a selected set of cryptographic parameters 104a, a source IP address and port number 203a received for message 203, a secure hash value over

PK.network 102a comprising H(PK.network 102a) 250, and identity for key server 102 comprising ID.key-server 102i, an ECC point value X1 211a, a server ephemeral public key E1 101a, a server ephemeral private key e1 101b, an ECC point value X2 212a, an ECC point value X3 215, a derived symmetric ciphering key K1 216a, and a second random number random2 205r. In exemplary
5 embodiments, the values depicted in the first row of server database 101d could comprise data recorded by a server 101 while conducting the series of steps for a step 222 and step 223 depicted and described in connection with Figure 2a above with a first device 103. The values depicted in the second row of server database 101d could comprise data recorded by a server 101 while conducting the series of steps for a step 222 and step 223 depicted and described in connection with Figure 2a
10 above with a second device 103, etc.

In exemplary embodiments for a server database 101d, a first device 103 could send server 101 a first value for device ephemeral public key Ed 103a, and the first value is depicted in Figure 2d as “103a-1”. Since server 101 could communicate with a plurality of devices 103, the second row in the depicted server database 101d could comprise data for the equivalent steps conducted with a
15 second device 103, such as recording a second value for device ephemeral public key Ed 103a for the second device. The second value for device ephemeral public key Ed 103a with the second device 103 is depicted in Figure 2d as “103a-2”. Equivalent notations for other keys or values are applicable as well, such as server database 101d recording a first secret X1 211a depicted as “211a-1” for a first device 103, and then recording a second secret X1 211a depicted as “211a-2”. Thus, as depicted a
20 server database 101d can record and operate with a plurality of different values for a key, where each are utilized by a different device. Although not depicted in Figure 2d, a server database could record device identity ID.device 103i as well. For embodiments where a device identity 103i is not available, then server 101 could keep track of different devices 103 for conducting the steps in Figure 2a by the source IP:port number 203a.

In some exemplary embodiments, a message 203 can include a secure hash value H(PK.network 102a) 250, as described for a message 203 in Figure 2a above. The receipt of a secure hash value H(PK.network 102a) 250 could be mapped to or associated with a key server 102 via a key server identity ID.key-server 102i, where the mapping of H 250 to ID.key-server 102i could be recorded in a server database 101d before device 103 sends a message 203. For these embodiments
30 and after receipt of message 203, server 101 could conduct a query of server database 101d using the received H 250 in a message 203 in order to select a key server 102 with ID.key-server 102i in order to send the message 206a to key server 102. In this manner, server 101 can communicate with a plurality of different key servers 102, and the destination of a message 206a (or key server 102) can be selected by the value H 250 received in a message 203. In other words, for a plurality of different
35 devices 103 communicating with a server 101, a first subset of devices 103 could record and use a first network static public key PK.network 102a, and a second subset of devices 103 could record and use

a second network static public key PK.network 102a. By receiving a value or identifier of the first or second key 102a in message 203 (such as H(PK.network 102a) 250), server 101 could use the data depicted for a server database 101d to select or identify the correct key server 102 in order to (i) send a message 206a and (ii) receive the correct secret X1 211a for the device 103 using a particular PK.network 102a.

Although the value H(PK.network 102a) 250 is depicted as recorded in a server database 101d in Figure 2d, a different value or identifier for the PK.network 102a could be recorded and utilized as well. In an exemplary embodiment, server 101 could receive the plaintext PK.network 102a in a message 203 and record the plaintext PK.network 102a in a server database 101d (instead of a hash value H 250). In another exemplary embodiment, an identity for key server 102 (such as ID.key-server 102i) could be selected or determined by server 101 using the selected set of cryptographic parameters 104a received in message 203 and recorded in a database 101d. For these embodiments, a first selected set of cryptographic parameters 104a could be associated with a first key server 102 (and first ID.key-server 102i) and a second set of cryptographic parameters 104a could be associated with a second key server 102 (and second ID.key-server 102i). Other possibilities exist as well for a server database 101d to record data in order to select a key server 102 for sending message 206a with device ephemeral public key Ed 103a based on data received in message 203, without departing from the scope of the present disclosure. As one example, the identity for key server 102 of ID.key-server 102i could be included in message 203 and the value for ID.key-server 102i could be recorded in a server database 101d by server 101.

In a server database 101d, although separate values are depicted for some data, such as values “102i-1” and “102i-2” for identities of key servers 102, some of the exemplary values can comprise identical strings or numbers. For example, data for two different devices 103 in a server database 101d could record the same name or value of “102i-2” for a single key server 102 to be associated with the two different devices 103. Likewise, two different devices 103 could share the same network static public key PK.network 102a, and thus H 250 can be the same value of an exemplary “250-2” for two different devices 103. A server database 101d could also record additional data and values than those depicted in Figure 2d for some exemplary embodiments. For example, server database 101d could record timestamps for when messages are transmitted or received, such that stale or data older than a specified range could be purged. Server database 101d could also record data received from device 103 in a message 210a, which could include data from a transducer operated by device 103.

Some data within a server database 101d could be recorded and operated on separately by server 101, such as server 101 not recording secrets such as X1 211a or X2 212a, etc. in a database 101d, but rather server 101 could record the values in volatile memory 101f of server 101. In exemplary embodiments, server database 101d could also operate in a distributed or “cloud”

configurations such that multiple different servers 101 could query and record data in server database 101d, where data for server database 101d is recorded in multiple, physically separated servers.

Cryptographic parameters 104 can specify sets of cryptographic parameters that are supported by server 101 in order to process message 203 and send response message 206c from Figure 2a.

5 Cryptographic parameters 104 can be recorded in a server database 101d, or in other locations within a system 100 and system 200. As depicted in Figure 1a, each of device 103, server 101, and key server 102 can record and operate with a set of cryptographic parameters 104. Cryptographic parameters 104 can record a collection of cryptographic algorithms or specifications such as a set identifier 104a, a key length 104b, an ECC curve name 104c, a hash algorithm 104d, symmetric ciphering key length
10 104e, settings for a symmetric ciphering algorithm 104f, and a random number length 104g.

As contemplated herein, when a selected set of cryptographic parameters such as using the words or description “parameters 104a” or “cryptographic parameters 104a” can specify a row of parameters or values in a set of cryptographic parameters 104, such that the collection of values in the row can be used with key pair generation functions 101x and 103x, ECDH key exchange 220, and
15 other cryptographic operations and steps as contemplated herein. Set identifier 104a can be an identity for a row or set of values for cryptographic parameters 104. For example, set “A” can comprise cryptographic suite 1 as specified in section 3.2.3 of DPP specification version 1.0. Key length 104b can be the length of keys in bits for PKI keys used in system 100 and system 200. ECC Curve name 104c can be a name for an ECC curve used with PKI keys and key exchange algorithms in system 100
20 and system 200.

Hash algorithm 104d in cryptographic parameters 104 can be the name of a secure hash algorithm, such as the exemplary SHA-256 algorithm depicted, which may also be referred to as “SHA-2”. Hash algorithm 104d can also be used in a key derivation function (e.g. KDF 216 above in Figure 2b and Figure 2c) and also with digital signature steps 207a and 209a. Settings for a symmetric
25 ciphering algorithm 104f can specify the identity or name of a symmetric ciphering algorithm 225 such as “AES”, “AES-SIV”, 3DES, Blowfish, etc. Random length 104g can specify the length in bits for random numbers or “nonces” generated by both device 103 and server 101, where the nonces can be used to prevent replay attacks and require messages transmitted and received to be unique. Other possibilities exist as well for data within cryptographic parameters 104, such as the specification of
30 point compression, encoding rules such as distinguished encoding rules (DER), ASN or CSN syntax notation, padding rules, etc.

Figure 2e

35 Figure 2e is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key using ECC point multiplication, in accordance with

exemplary embodiments. An ECDH key exchange step 218n can be conducted by a device 103, and use the steps for an ECDH key exchange step 218, with the additional steps of conducting an ECC point multiplication using numbers N1 298 and N2 299. A key derivation step 213n can be conducted by a server 101, and use the steps for a key derivation step 213, with the additional steps of conducting
5 an ECC point multiplication using the same numbers N1 298 and N2 299. In other words, (i) ECDH key exchange step 218 can comprise the depicted ECDH key exchange step 218n where the numbers for N1 298 and N2 299 are equal to the value of “1”, and (ii) key derivation step 213 can comprise the depicted key derivation step 213n where the numbers for N1 298 and N2 299 are also equal to the value of “1”. In some exemplary embodiments, (i) an ECDH key exchange step 218 depicted and
10 described in connection with Figure 2a for device 103 can comprise the ECDH key exchange step 218n with point multiplication, and key derivation step 213 for server 101 can comprise the key derivation step 213n with point multiplication. The set of parameters 104a from figures above, such as with Figure 2a, can be used with both ECDH key exchange step 218n and key derivation step 213n.

A device 103 can conduct a key exchange step 218n. At step 218n, a device 103 can conduct
15 a first ECDH key exchange step 220 and a second ECDH key exchange step 220. For a step 218n, a first ECDH key exchange step 220 can be conducted by device 103 with (i) the server ephemeral public key E1 101a received in a message 206c from Figure 2a and (ii) the recorded device ephemeral private key ed 103b, and the resulting point multiplied by the number N1 298. Note that the ECC point resulting from the first ECDH key exchange 220 in the previous sentence will also equal the
20 point X2 212a multiplied by the number N1 298, where the calculation of point X2 212a is depicted and described in connection with a key exchange step 212 in Figure 2b.

Continuing with step 218n, a device 103 can conduct the second ECDH key exchange step 220 with (i) the network static public key PK.network 102a recorded in device 103 and (ii) the recorded device ephemeral private key ed 103b, and the resulting point multiplied by the number N2 299. Note
25 that the ECC point resulting from the second ECDH key exchange 220 in the previous sentence will also be equal to the point X1 211a multiplied by the number N2 299, where the calculation of point X1 211a is depicted and described in connection with key exchange step 211 in Figure 2b.

Continuing with step 218n, a device 103 can conduct an ECC point addition operation on the two points resulting from (i) the first ECDH key exchange step 220 multiplied by N1 298 and (ii) the
30 second ECDH key exchange step multiplied by N2 299. In other words, a device 103 can conduct an ECDH point addition operation with (i) the value X2 212a multiplied by N1 298 and (ii) the value X1 211a multiplied by the value N2 299, in order to derive a secret X3' 215a that is mutually shared with server 101.

Exemplary data and numbers can be provided to demonstrate the calculations for (i) key
35 exchange step 218n and (ii) key derivation step 213n. The exemplary data can comprise decimal numbers for the example ECC PKI keys and exchanged keys described above in Figure 2b. The first

ECDH key exchange 220 for device 103 using (i) the exemplary numerical value for device ephemeral private key ed 103b in Figure 2c and (ii) the exemplary numerical value for server ephemeral public key E1 101a in Figure 2c, using parameters 104a, will result in the exemplary number or value for secret X1 211a, where parameters 104a can comprise the elliptic curve of “secp256r1” with key lengths of 256 bit long keys:

X:

11490047198680522515311590962599671482029417064351337303313906642805743573119

Y:

27933966560238204731245097943399084523809481833434754409723604970366082021855

For an exemplary value of “3” for N1 298, the resulting ECC point multiplication of X1 211a by N1 298 with the value of “3” will result in the following point “3 x X1”:

X:

60742753813277956134086722801387134015749233649228884236187651653814176225536

Y:

58611335288463132268275870174894337145888786863441350683708443176926328298969

The second ECDH key exchange 220 for device 103 in a step 218n using (i) the exemplary numerical value for device ephemeral private key ed 103b in Figure 2c and (ii) the exemplary numerical value for network static public key PK.network 102a in Figure 2c, using parameters 104a, will result in the exemplary number or value for secret X2 212a:

X:

78944719651774206698250588701582570633503182903415394243006529481189158194650

Y:

11227712702924684581834935828837489140201820424536062912051086382324589445237

For an exemplary value of “7” for N2 299, the resulting ECC point multiplication of X2 212a by N2 299 with the value of “7” will result in the following point “7 x X2”:

X:

97872096638582215727304642389226702208575594850473136075994007337240867556563

Y:

30901113762050629628611789412759390525616003079040872429940997779854500728255

An ECC point addition for the two points “3 x X1” and “7 x X2” will result in the following point, which can equal the shared secret X3’ 215a for a key exchange step 218n:

5 X:

107460308686621111684900795619695874701132258776388121688297958325813410507748

Y:

104797039912644919810998853512360434930336867141382017165496514798694755489900

10 The above values for N1 298 and N2 299 are exemplary, and any numeric value less than the large prime number p for a named elliptic curve could be selected for both N1 298 and N2 299.

Continuing with step 218n, derived shared secret key X3’ 215a can be input into a key derivation function 216 where the key derivation function 216 can be equivalent to the key derivation function 216 depicted and described in connection with Figure 2b above for a key derivation step 213.

15 Note that for key derivation steps in the present disclosure, the X coordinate of a derived public key can be taken or used as input into the key derivation function. The output of a key derivation function 216 can comprise both (i) a symmetric ciphering key K1 216a and (ii) a MAC key 216b. The use of key K1 216a and MAC key 216b are described in connection with encryption step 217 and decryption step 219 in Figure 2c.

20 For a key derivation step 213n by server 101, server 101 can conduct the equivalent steps as key derivation step 213 in Figure 2b, with point multiplication operations depicted in Figure 2e. Server 101 can perform an ECC point addition and point multiplication step 214a using the values X1 211a and X2 212a, as well as the numbers N1 298 and N2 299. The value X1 211a could be received by server 101 from key server 102 in message 206a. Note that the value X1 211a is derived by key server
25 102 using an ECDH key exchange step 211 as depicted and described in connection with Figure 2b. A server 101 could calculate the value for X2 212a using an ECDH key exchange step 212 in Figure 2b. The value or point X1 211a can be multiplied by number N2 299. The value or point X2 212a can be multiplied by the number N1 298. An ECC point addition can be performed on the two ECC points obtained in each of the previous two sentences in order to calculate a value X3’ 215a. The
30 exemplary calculations for point multiplication on X1 211a (with N2 299) and X2 212a (with N2 298) by device 103 would also be calculated by server 101. In other words, the exemplary data and numbers depicted above for the calculations by device 103 could also be calculated by server 101 in order to mutually derive the same value for X3’ 215a. The mutually derived value for X3’ 215a can be input into key derivation function 216 in order to calculate a symmetric ciphering key K1 216a and a MAC
35 key 216b, which can comprise the same numbers as calculated by device 103 in a step 218n.

The source of values for N1 298 and N2 299 for both device 103 and server 101 could be mutually obtained in several ways. N1 298 and N2 299 could be recorded and shared with a set of cryptographic parameters 104, such that selecting a subset of the cryptographic parameters 104a could determine the values or numbers to use for N1 298 and N2 299. In another exemplary embodiment, N1 298 and/or N2 299 could comprise pre-shared secret values or keys, such that device 103 receives the values in a secure manner before sending message 203, such as, but not limited to, recording the values at functionally the same time network static public key PK.network 102a is recorded in device 103. Server 101 could receive the values N1 298 and N2 299 in a secure manner, such as from key server 102 in a secure session 221. Other possibilities exist as well for a device 103 and a server 101 to obtain the numbers N1 298 and N2 299 without departing from the scope of the present disclosure. In exemplary embodiments, the number for N1 298 or N2 299 can be either equal, or the numbers could comprise different values.

A device 103 and a server 101 could also conduct a number derivation step 297 in order to obtain the numbers N1 298 and N2 299, which is also depicted in Figure 2e. For a number derivation step 297, a static public key can be input into a secure hash algorithm 291, such as SHA-256. The static public key can be any public key shared between a device 103 and server 101 (e.g. where one node records the public key and the other node records the corresponding private key). In exemplary embodiments depicted in Figure 2e, the public key for a number derivation step 297 can comprise the network static public key PK.network 102a, where a server 101 can derive or calculate the network static public key can be derived from the network static private key SK.network 102b using parameters 104. Other exemplary public keys shared between device 103 and server 101 can comprise any of public keys Ed 103a, E1 101a, D1 103c, etc. The node recording the corresponding private key can calculate the public key using the parameters.

The output of the secure hash algorithm 291 can be input into a select digits function 292. The select digits function 292 could take a subset of the hash value resulting from hash 291, such as leading digits for N1 298 and trailing digits for N2 299. Or, a number N1 298 could be derived from a select digits function 292 over a hash 291 of the X coordinate of a public key and the number N2 299 could be derived by a select digits function 292 over a hash 291 of the Y coordinate of the same public key. Other subsets or logic for the select digits function 292 using the hash value from hash algorithm 291 can be used as well, without departing from the scope of the present disclosure. The output of the select digits function 292 can comprise the value N1 298 and N2 299. Since both device 103 and server 101 and/or network 105 can securely share PK.network 102a, then the same calculations for a number derivation step 297 can be performed by the nodes in order to mutually obtain the numbers N1 298 and N2 299. The values for N1 298 and N2 299 can be used by (i) device 101 when conducting the key exchange step 218n and (ii) server 101 when conducting the key derivation step 213n.

Figure 3a

Figure 3a is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a mobile device, a g node b, and a key server, in accordance with exemplary embodiments. System 301 can include a mobile device 103', a "next generation node b" 101', and a key server 102. Mobile device 103' can comprise a smart phone, a device for the "Internet of Things" (IoT), a tablet with a modem, or possibly a fixed station device that connects with a 5G or 6G network. Mobile device 103' can operate similar to a device 103, with the additional functionality of connecting to a wireless network, where the network supports 3GPP standards and can also comprise a wide area network such as a public land mobile network. A "next generation node b" 101' (depicted as gNb 101') can contain the equivalent electrical components as those depicted for a server 101 in Figure 1b, except gNb 101' can also operate as a base transceiver station to send and receive data wirelessly with mobile device 103'. The key server 102 could operate as part of an Authentication Server Function (AUSF) or equivalent functionality. Note that the distributed nature of the ECDH key exchanges as depicted in Figure 2a and Figure 2b and Figure 2c have benefits for the wireless WAN architecture in Figure 3a, SK.network 102b for a mobile device 103' does not need to be recorded or operated by a gNb 101'.

In exemplary embodiments, a mobile device 103', a gNb 101', and a key server 102 can conduct a step 222', where a step 222' can comprise primarily the step 222 as depicted and described in Figure 2a. There can be some differences between a step 222 and a step 222'. Note that before the steps 222' depicted in Figure 3a, a mobile device 103' and a gNb 101' could conduct steps to establish communications between the nodes, such as recording parameters for RF communications by the mobile device 103' in a SIM card or eUICC. A mobile device 103' could also conduct steps to authenticate the network 105 operating a gNb 101'. For a step 222', a mobile device 103' can send message 203 with the device ephemeral public key Ed 103a and also an obfuscated identity for device 103', where the obfuscated identity can also comprise a temporary identity for device 103. A gNb 101' can use the obfuscated identity to track the device 103 from a potential plurality of devices 103 communicating over a wireless network.

The gNb 101' can forward the device identity and the received device ephemeral public key to the key server 102. The key server 102 can look up a unique key 102v for device 103 for the network static private key 102b corresponding to the network static public key 102a recorded by the device 103. The key server 102 can calculate value X1 211a as depicted in Figure 2b, and send the gNb 101' the value X1 211a over a secure session. The gNb 101' can conduct an ECDH key exchange step 212 and calculate value X2 212a, using the received device ephemeral public key Ed 103a and the derived server ephemeral private key e1 101b. The gNb 101' can calculate the value X3 215 via ECC point addition over X1 211a and X2 212a. The gNb 101' can calculate a symmetric ciphering

key K1 216a using the value X3 215 and a KDF 216. The gNb 101' can send the mobile device 103' the derived server ephemeral public key E1 101 in a message 206c from a step 222. Note that some data within ciphertext 217b can be omitted from a message 206c in a step 222', where step 222' is depicted in Figure 3a and comprises equivalent steps as a step 222 in Figure 2a.

5 The mobile device 103' can receive the message 206c from a step 222'. The mobile device 103', gNb 101', and key server 102 can conduct a step 223, where a step 223 was depicted and described in connection with Figure 2a above. The mobile device 103' can send gNb 101' a message 210a with ciphertext 217d, where ciphertext 217d can include a device identity ID.device 103i as plaintext encrypted in the ciphertext 217d. The ciphertext 217d can be encrypted with the derived
 10 symmetric ciphering key K1 216a and a symmetric ciphering algorithm 225, where key K1 216a was derived by mobile device 103' in a step 222'. The identity for the mobile device 103i can comprise a subscription permanent identifier (SUPI), and by transmitting the SUPI within a ciphertext 217d, the SUPI can remain confidential and not transmitted in the clear through a wireless network. Other possibilities for the use of a step 222' and a step 223 between a mobile device 103' and gNb 101' exist
 15 without departing from the scope of the present disclosure.

Figure 3b

Figure 3b is a simplified message flow diagram illustrating an exemplary system with
 20 exemplary data sent and received by a client, a server, and a key server, in accordance with exemplary embodiments. System 302 can include a client 103', a server comprising server 101, and a key server 102. In exemplary embodiments, client 103' can comprise a client using security steps as described in by transport layer security (TLS) sessions version 1.3 and also subsequent and related versions of IETF RFC standards. Client 103' can also comprise a client using security steps as described in
 25 datagram transport layer security (DTLS) RFC 6347 and subsequent versions that incorporate ECDH key exchanges. Although depicted in Figure 3b as a client 103', the client 103' could also comprise a device 103, where the device 103 can conduct the steps of a client 103' at the networking, transport, and application layer of the traditional Open Systems Interconnection (OSI) model.

Client 103' can comprise a computing device that records a network static public key
 30 PK.network 102a. Note that TLS version 1.3 and DTLS version 1.3 contemplate that the client and a server can use ephemeral ECDH key exchanges (one on the client and one on the server) in order to establish a mutually derived secret shared key for a symmetric ciphering algorithm. The difference between (i) a client 103' (which can comprise a device 103 supporting TLS or DTLS standards) and (ii) a client for TLS or DTLS standards can be that client 103' can record a network static public key
 35 PK.network 102a. As depicted in Figure 1c, the network static public key PK.network 102a could comprise either (i) a shared key 102z across a plurality of different devices 103 (or clients 103'), or

(ii) a unique key 102v, where the network static public key PK.network 102a is a unique number or string or point for client 103'. The key PK.network 102a could be received by client 103' in a secure manner before a client 103' conducts a step 222 with server 101. In exemplary embodiments, PK.network 102a could be received in the form of a certificate with PK.network 102a from a prior
5 TLS or DTLS session before client 103' begins the TLS or DTLS session depicted in Figure 3b. Or, PK.network 102a could be recorded with a set of certificate authority certificates stored with installation of an operating system for device 103.

The use of a network static public key PK.network 102a by client 103' in a step 222 to conduct an ECDHE key exchange with server 101 can have many benefits. The standard handshake as
10 currently proposed for TLS version 1.3 as of June 2018 assumes that a client 103' and a server 101 have no prior relationship. However, for many instances of communication between a client 103' and a server 101, the client 103' may have previously communicated with another server on a network 105 other than server 101. For example, with web browsing a web browser client such as a client 103' will often revisit the same web sites over time, such as a first web site for social networking, a second
15 web site for a search engine, a third web site for news, etc. A TLS or a DTLS session could utilize the fact that the same sites are often re-visited in order to increase security, using the depicted steps of 222 and 223 for a client 103', server 101, and key server 102. Steps 222 in Figure 3b can comprise the set of steps 222 depicted and described in connection with Figure 2a, and steps 223 in Figure 3b can also comprise the set of steps 223 depicted and described in connection with Figure 2a.

Before conducting step 222 in Figure 3b, a client 103' could receive key PK.network 102a
20 from another server in network 105, such as a different web server providing functionality equivalent to server 101. PK.network 102a could also be stored or recorded by a client 103' along with a set of certificate authority certificates (including root certificates) for an operating system of a device operating the client 103'. Or, PK.network 102a could be securely received in a previous TLS or DTLS
25 session, such as receiving PK.network 102a in a certificate verified by client 103' before client 103' conducts a step 222 in Figure 3b. The certificate could be verified by client 103' using a certificate authority root certificate, including verification through any intermediate certificate authority certificates. The client 103' could record the network static public key PK.network 102a in a table 103t along with parameters 104a associated with PK.network 102a. In exemplary embodiments, a
30 table 103t could include certificates such as X.509 v3 certificates for the network static public keys PK.network 102a, where the certificates include digital signatures from a certificate authority. The key PK.network 102a could also be recorded with a URL or domain name (e.g. a server name indication), such that the client 103' would use the key PK.network 102a when establishing a subsequent TLS or DTLS session with server 101, where server 101 uses the recorded URL or domain
35 name. Further, server 101 could be configured so that any key Ed 103a received from IP network 107 on an IP address and/or port number used by server 101 would be forwarded to key server 102, where

key server 102 could record and operate with the SK.network 102b corresponding to the public key for PK.network 102a recorded by client 103'. Server 101 could also operate such that a URL is associated with a key server 102 and/or PK.network 102a, such that a call or request of the URL could be used to select the key server 102 and/or PK.network 102a.

5 For a step 222, a client 103' can (i) derive a device ephemeral public key Ed 103a and private key ed 103b using parameters 104a stored with PK.network 102a and (ii) send server 101 a message 203. The message 203 can include the key Ed 103a and the set of cryptographic parameters 104a associated with Ed 103a. In some exemplary embodiments client 103' implements TLS or DTLS, and message 203 can optionally omit a device identity ID.device 103i. Server 101 could operate in a
10 manner such that (i) Ed 103a is forwarded to key server 102, and (ii) server 101 derives an ephemeral PKI key pair. Key server 102 can conduct an ECDHE key exchange as depicted for a step 222 in Figure 2a using a step 211 in order to calculate the secret value X1 211a. Key server 102 can send server 101 the value X1 211a. Server 101 can use the value X1 211a, along with the derivation of a
15 second secret X2 212a in order to calculate a symmetric ciphering key K1 216a, using the key derivation step 213 with ECC point addition 214 over X1 211a and X2 212a. Thus, by using the embodiment depicted in Figure 3b, a transport layer security session can have security increased, where (a) the ECDHE key exchange contemplated by TLS v1.3 (which would be key exchange 212 in Figure 2b) can also add (b) the additional key exchange step 211a by a key server 102. Note that the mutual derivation of symmetric ciphering key K1 216a by client 103' and server 101 can comprise
20 a one-way authentication of server 101, since server 101 can only derive the key K1 216a if server 101 operates in a network 105 that also records and operates with key SK.network 102b.

The server 101 can send the client 103' the derived server ephemeral public key E1 101a in a message 206c from a step 222. Key E1 101a could be derived by a step (ii) in the above paragraph. Message 206c could comprise a "Server Hello" according to TLS v1.3 in the document "draft-ietf-tls-
25 tls13-28". The ciphertext in the Server Hello can be ciphertext 217b as depicted in Figure 2a, where the ciphertext 217a is encrypted with the mutually derived symmetric ciphering key K1 216a. Note that a step 222 for Figure 3b increases security for a TLS session, since an active attacker could operate as a "man in the middle" between a real client or "true client" and the server 101, where the "man in the middle" could derive its own key Ed 103a and substitute that for the real key Ed 103a from the
30 real client or "true client". Without use of a PK.network 102a, a "man in the middle" (deriving and substituting a key Ed 103a) could (a) mutually derive a symmetric ciphering key similar to K1 216a with server 101 and then (b) receive and decrypt the ciphertext 217b. However, the use of PK.network 102a can stop a "man in the middle" attack since a "man in the middle" cannot derive key K1 216a without also recording the SK.network 102b, which can remain secret and not available to the "man
35 in the middle".

The client 103' can receive the message 206c from a step 222 from a server 101. The client 103', server 101, and key server 102 can conduct a step 223, where a step 223 was depicted and described in connection with Figure 2a above. The client 103' can derive the same key K1 216c using a step 218 and the PK.network 102a. The client 103' can decrypt ciphertext 217b using key K1 216a. The client 103' can process the plaintext data, such as recording a certificate for server 101 (e.g. cert.server 101c from Figure 2a), and verifying a signature 101s from server 101. The client can also read a random number transmitted in the ciphertext 217b and create a digital signature over the random number. The client can encrypt a ciphertext 217d with data to respond to server 101. The ciphertext 217d can be encrypted with the derived symmetric ciphering key K1 216b and a symmetric ciphering algorithm 211a, where key K1 216a was derived by client 103' in a step 223. Other possibilities exist for the use of a step 222 and a step 223 between a client 103' and server 101 without departing from the scope of the present disclosure.

For the exemplary embodiment depicted in Figure 3b for support of TLS and DTLS secured data sessions, a message 203 can comprise a "client hello" message, a message 206c can comprise a "server hello" message, and message 210a can comprise a "finished" message from the client 103'. For exemplary embodiments, message 203 as a "client hello" message can omit a device identity 103i (such as a permanent identifier for client 103' or device 103, but the "client hello" message could include other identifying information for client 103' such as (i) an originating IP address and source port number for message 203, (ii) an obfuscated and/or temporary identity such as a random number for a session, and other possibilities exist as well without departing from the scope of the present disclosure.

In addition, embodiments depicted in Figure 3b solve a significant challenge for resource constrained devices to fully authenticate a certificate cert.server 101c. There could be many layers of intermediate certificates between cert.server 101c and a certificate authority root certificate stored in device 103. Checking for certificate validity for all intermediate certificates and for revocation or OSCP signatures and/or stapling could add many levels of signature verifications. ARM reported a 32 Cortex M4 processor with 32 bits and operating at 84 Mhz requires ~420 ms for a single ECDSA signature verification (secp521r1) ("Performance of State-of-the-Art Cryptography on ARM-based Microprocessors", July 21, 2015). There could be 8 or more signatures to be verified for a full certificate chain verification of cert.server 101c and related OSCP signatures. A device could conduct the single authenticated key exchange step 218 in less than 15% of the time and power required for the full, traditional certificate chain verification. Also, there are reduced chances for errors due to unsupported parameters for (x) a single authenticated ECDH key exchange step compared to (y) multiple certificate verifications steps with OSCP verification. Consequently, the communications for a TLS session or DTLS session can remain secured more efficiently using a step 222 and step 223, while recording and using (i) SK.network 102b with network 105 and (ii) PK.network 102a with client

103', compared to traditional TLS or DTLS implementations with multiple layers of certificate authorities through root certificates.

5 **Figure 3c**

Figure 3c is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by an initiator, a responder, and a key server, in accordance with exemplary embodiments. System 303 can include an initiator 103', a responder 101' and a key server 102. Initiator 103' can comprise a computing device 103, with the specific additional functionality of an initiator according to the DPP Specification Versions 1.0 from the WiFi Alliance. Responder 101' can comprise a device with (i) electrical components similar or equivalent to a server 101 depicted in Figure 1b above, and (ii) the specific additional functionality of a responder according to the DPP Specification Version 1.0 of the WiFi Alliance. For example, initiator 103' and responder 101' can communicate via a WiFi network on a LAN between the two devices, which could also comprise the IP network 107. Responder 101' can operate in a networked configuration to communicate with key server 102 via a private network 107a or a secure session 221 as depicted in Figure 2a. In some embodiments, responder 101' can communicate with key server 102 via an IP network 107, where the use of secure session 221 can create a private network 107a between responder 101' and key server 102.

An initiator 103', responder 101' and a key server 102 can conduct a step 222, where a step 222 is depicted and described in connection with Figure 2a above. An initiator 103', responder 101' and a key server 102 can then conduct a step 223, where a step 223 is depicted and described in connection with Figure 2a above. As depicted in Figure 3c, several PKI keys within a DPP specification version 1.0 can have corresponding keys for a step 222 and step 223. Note that additional steps in addition to those depicted in Figure 3c can be conducted by an initiator 103' and a responder 101', such as responder 101' deriving PKI keys in a step 101x from Figure 1a and also conducting additional ECDH key exchanges in order to derive a symmetric ciphering key ke in addition to symmetric ciphering key K1 216a. In other words, initiator 103' and responder 101' could perform additional ciphering than that depicted for a step 222 in Figure 2a, but for exemplary embodiments such as that depicted in Figure 3c the initiator 103' and responder 101' could conduct at least the steps depicted in order to mutually derive a symmetric ciphering key K1 216a and use the key to create a ciphertext 217b by responder 101' and decrypt the ciphertext 217b by initiator 103'.

As depicted in Figure 3c, the device ephemeral public key Ed 103a can comprise the initiator protocol public key Pi 303a. The device ephemeral private key ed 103b can comprise the initiator protocol private key pi 303b. The server ephemeral public key E1 101a can comprise the responder

protocol public key Pr 301a. The server ephemeral private key e1 101b can comprise the responder protocol private key pr 301b. The network static public key PK.network 102a can comprise the responder bootstrap public key 302a. The network static private key SK.network 102b can comprise the responder bootstrap private key 302b. As described below, other steps from Figure 2a can be equivalent to those depicted in Figure 3c.

For a message 203 sent from initiator 103' to responder 101', the message 203 with the key Pi 303a can also include a ciphertext. The message 203 in a step 222 can comprise a "DPP Authentication Request" message from the DPP v1.0 standard. Responder 101' can communicate with key server 102 and receive the value X1 211a. Responder 101' can also derive the server ephemeral public key E1 101 (comprising the responder protocol public key Pr 301a) and the server ephemeral private key e1 101b (comprising the responder protocol private key pr 301b). The Responder 101' can use KDF 216 to convert X1 211a into a symmetric encryption key (which can be different than key K1 216a from Figures above). Responder 101' can use the symmetric encryption key from X1 211a to decrypt the ciphertext with a message 203. Responder 101' can then conduct the key exchange step 212 and step 213, along with modified versions of KDF 216 in order to derive a key ke. Responder 101' can encrypt data with the key ke and send initiator 103' a message 206c with the encrypted data. The message 206c can comprise a "DPP Authentication Response" message from the DPP v1.0 standard. Initiator 103' can then send responder 101' a "DPP Configuration Request" message, which could comprise message 210a in a step 223 as depicted in Figure 2a.

A benefit for the use of a step 222 and step 223 for an initiator 103' and a responder 101' is that the responder bootstrap private key br 302b can remain securely recorded in a network 105 and does not need to be recorded and operated by responder 101. In this manner, the responder bootstrap public key Br 302a can be freely shared with multiple different initiators 103', including recording the key Br 302a in a plurality of initiators 103' in the form of a shared key 102z as depicted in Figure 1c. The use of a shared key 102z with multiple different initiators 103' (while keeping SK.network 102b or key br 302b securely recorded in a key server 102) simplifies the distribution of key Br 302a to multiple different initiators 103'.

For exemplary embodiments, the initiators 103' could have a key Br 302a recorded during manufacturing or distribution of the computing device operating initiator 103'. In other words, a device manufacturer upon device manufacturing with initiator 103' may not know which responder 101' may communicate with initiator 103' during a subsequent DPP session. However, a manufacturer of device with initiator 103' could record a plurality of different keys Br 302a for different networks 105 (similar to different keys PK.network 102a in for a table 103t Figure 1c), and in this manner initiator 103' can have a higher probability of successfully using a pre-recorded key Br 302a (or key PK.network 102a) in order to conduct a DPP session without requiring a separate or different additional step of acquiring the key Br 302a "out of band". Thus, the use of the embodiment

for an initiator 103' and a responder 101' can simplify the use and deployment of DPP sessions, while simultaneously increasing the securing of the session, since the responder bootstrap private key br 302b (in the form of SK.network 102b) can remain securely recorded within a network 105 on a key server 102.

5

Conclusion

Various exemplary embodiments have been described above. Those skilled in the art will understand, however, that changes and modifications may be made to those examples without departing from the scope of the claims.

CLAIMS

1. A method for securing communications, the method performed by a set of servers, the method comprising:

- a) receiving, by a first server and from a device, a first message comprising a device ephemeral public key, wherein the device ephemeral public key is associated with a set of cryptographic parameters;
- b) deriving, by the first server, a server ephemeral public key and a corresponding server ephemeral private key;
- c) receiving, by a second server, the device ephemeral public key from the first server;
- d) conducting, by the second server, a first elliptic curve Diffie Hellman (ECDH) key exchange using at least (i) the device ephemeral public key and (ii) a network static private key in order to derive a first shared secret;
- e) sending, by the second server and to the first server, the first shared secret over a secure connection;
- f) conducting, by the first server, a second ECDH key exchange using at least, (i) the received device ephemeral public key and (ii) the server ephemeral private key in order to derive a second shared secret;
- g) conducting, by the first server, an ECC point addition operation using the first shared secret and the second shared secret in order to derive a third shared secret;
- h) deriving, by the first server, a symmetric ciphering key using at least (i) a key derivation function and (ii) the third shared secret;
- i) generating, by the first server, a ciphertext using at least the symmetric ciphering key; and
- j) sending, by the first server and to the device, a second message with the server ephemeral public key and the ciphertext.

2. The method of claim 1, wherein the set of servers comprises the first server and the second server, and wherein the first server and the second server record the set of cryptographic parameters before the first server receives the first message.

3. The method of claim 2, further comprising recording, by the second server, the network
5 static private key before the first server receives the first message.

4. The method of claim 1, further comprising recording, by the device before step a), (i) a network static public key corresponding to the network static private key and (ii) the set of cryptographic parameters.

5. The method of claim 4, further comprising conducting, by the device, a third ECDH
10 key exchange using at least (i) a device ephemeral private key corresponding to the device ephemeral public key, (ii) the server ephemeral public key, and (iii) the network static public key in order to derive the third shared secret.

6. The method of claim 5, further comprising conducting, by the device, a second ECC
15 point addition operation with the device ephemeral public key and the network static public key in order to derive the third shared secret, wherein the device derives the symmetric ciphering key using at least (i) the key derivation function and (ii) the third shared secret.

7. A method for conducting a secure session between a device and a server, the method performed by the device, the method comprising:

a) receiving, by the device, a certificate for a network public key with a set of
20 cryptographic parameters, wherein the device verifies the certificate with at least a root certificate stored by the device.

b) deriving, by the device, a device ephemeral public key and a corresponding device ephemeral private key using at least the set of cryptographic parameters;

c) sending, by the device and to the server, the device ephemeral public key and the set of
25 cryptographic parameters;

d) receiving, by the device and from the server, a server ephemeral public key and a ciphertext;

e) conducting, by the device, an elliptic curve Diffie Hellman (ECDH) key exchange using at least (i) the server ephemeral public key, (ii) the network public key, and (iii) the device ephemeral private key in order to derive a shared secret;

h) deriving, by the device, a symmetric ciphering key using at least (i) a key derivation function and (ii) the shared secret; and

i) decrypting, by the device, the ciphertext using the symmetric ciphering key.

8. The method of claim 7, wherein the device conducts the ECDH key exchange by conducting (i) an ECC point addition operation with the server ephemeral public key and the network public key to derive a point, and (ii) an ECC point multiplication operation with the point and the device ephemeral private key.

9. The method of claim 7, wherein the device conducts the ECDH key exchange by conducting (i) a first ECC point multiplication operation with the server ephemeral public key and the device ephemeral private key to derive a first point, (ii) a second ECC point multiplication operation with the network ephemeral public key and the device ephemeral private key to derive a second point, and (iii) an ECC point addition operation with the first point and the second point to derive the shared secret.

10. The method of any one of claims 7-9, further comprising: conducting, by the server, a second ECDH key exchange with (i) the received device ephemeral public key and (ii) a server ephemeral private key in order to derive a second shared secret; conducting, by a key server, a third ECDH key exchange with (i) the device ephemeral public key and (ii) a network static private key in order to derive a third shared secret; sending, by the key server and to the server, the third shared secret; and conducting, by the server, an ECC point addition operation with the second shared secret and the third shared secret in order to derive the shared secret.

11. The method of any one of claims 7-10, further comprising before step b), (i) storing a network public key table with a plurality of network public keys and associated domain names, and (ii) selecting the network public key from the network public key table with a domain name for the server.

5 12. The method of any one of claims 7-11, wherein the network public key comprises a unique key for the device, and wherein the device sends the server a secure hash value for the network public key with the device ephemeral public key.

13. The method of claim 7, wherein the network public key comprises a shared key, wherein the server selects a key server using a domain name for the server in the set of
10 cryptographic parameters, wherein the key server stores a network private key for the network public key.

14. A method for a key server to support an authenticated key exchange, the method performed by the key server, the method comprising:

a) recording in a database (i) a plurality of network private keys with parameters and (ii) a
15 plurality of secure hash values for corresponding network public keys;
b) establishing a secure session with a server, wherein the server communicates with a plurality of devices over a public Internet;
c) receiving, from the server, a device ephemeral public key and a secure hash value;
d) selecting, from the database, a network private key and the parameters for the network
20 private key using the secure hash value;
e) conducting a first elliptic curve Diffie Hellman (ECDH) key exchange using the network private key, the device ephemeral public key, and the parameters in order to derive a point; and
f) sending, to the server, the derived point, wherein the derived point is used with a second
25 ECDH key exchange and a server ephemeral private key in order to derive a shared secret and a symmetric ciphering key.

15. The method of claim 14, further comprising: conducting, by the server, an ECC point addition operation with the derived point and a value from the second ECDH key exchange in order to derive the shared secret.

5 16. The method of any one of claims 14-15, further comprising: conducting, by a device, a third ECDH key exchange in order to derive the shared secret, and wherein the third ECDH key exchange uses at least (i) a received server ephemeral public key for the server ephemeral private key, (ii) a stored network public key for the network private key, and (iii) the device ephemeral private key.

10 17. The method of claim 16, further comprising: establishing, by the device and the server a secure session over the public Internet using at least the symmetric ciphering key.

18. The method of claim 16, wherein the network private key comprises a unique key for the device, and wherein the device receives the network public key in a certificate.

15 19. The method of any one of claims 14-18, further comprising, after step d) and before step e), verifying, by the key server, the received device ephemeral public key with the selected set of cryptographic parameters.

20. The method of any one of claims 14-19, further comprising, after step f), receiving, from the server, a message after the server decrypts a ciphertext with the symmetric ciphering key.

Figure 1a

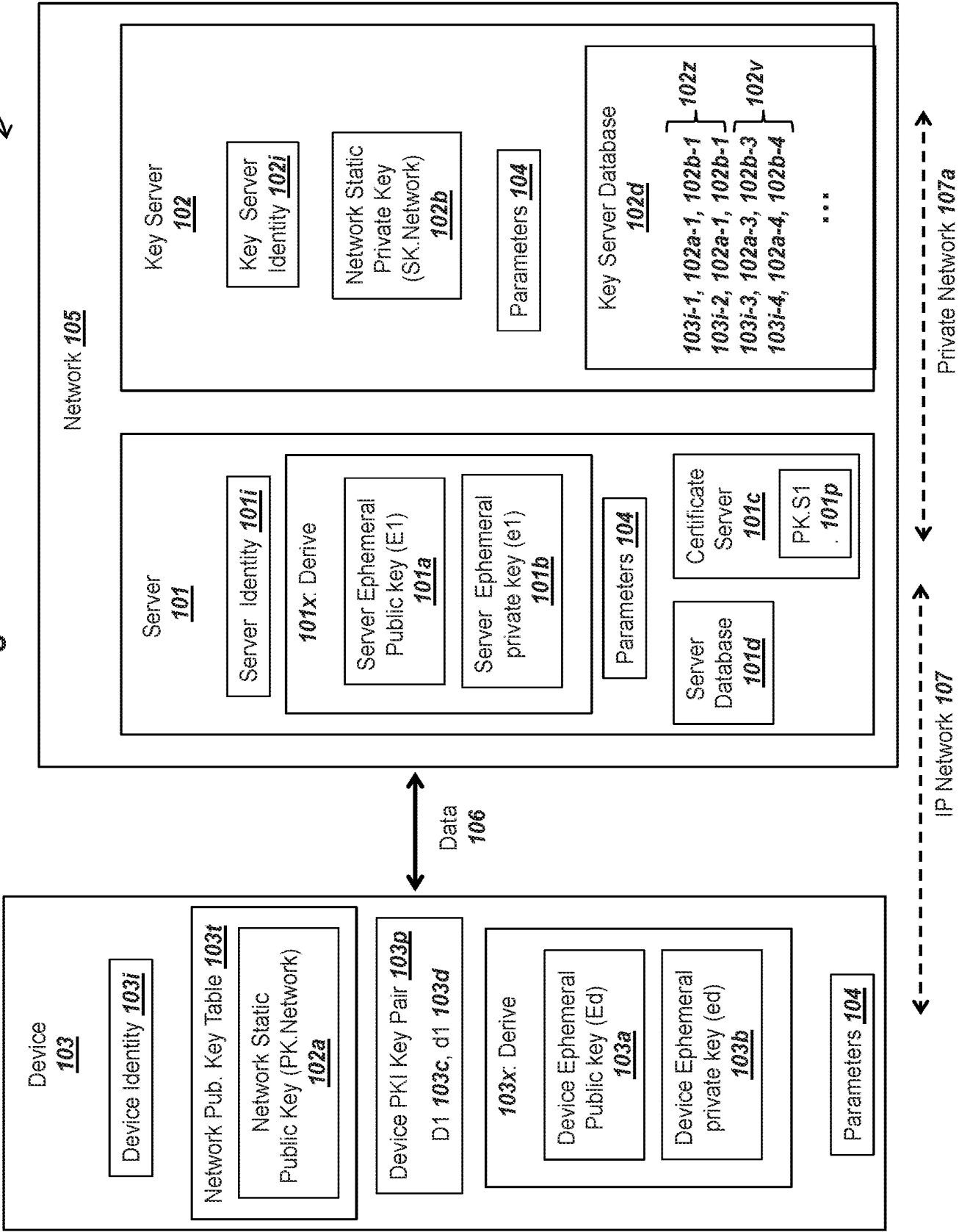


Figure 1b

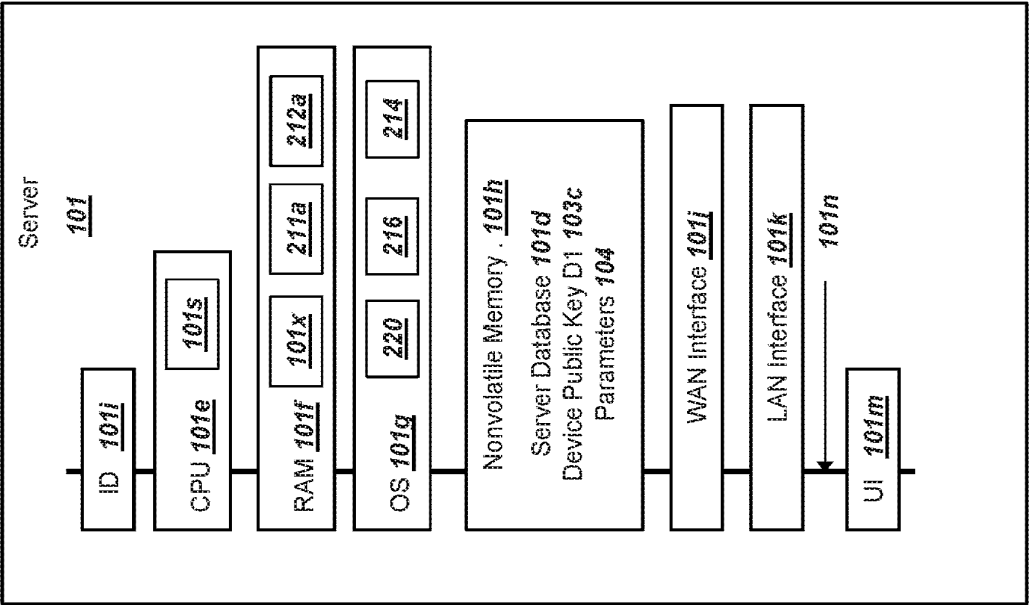
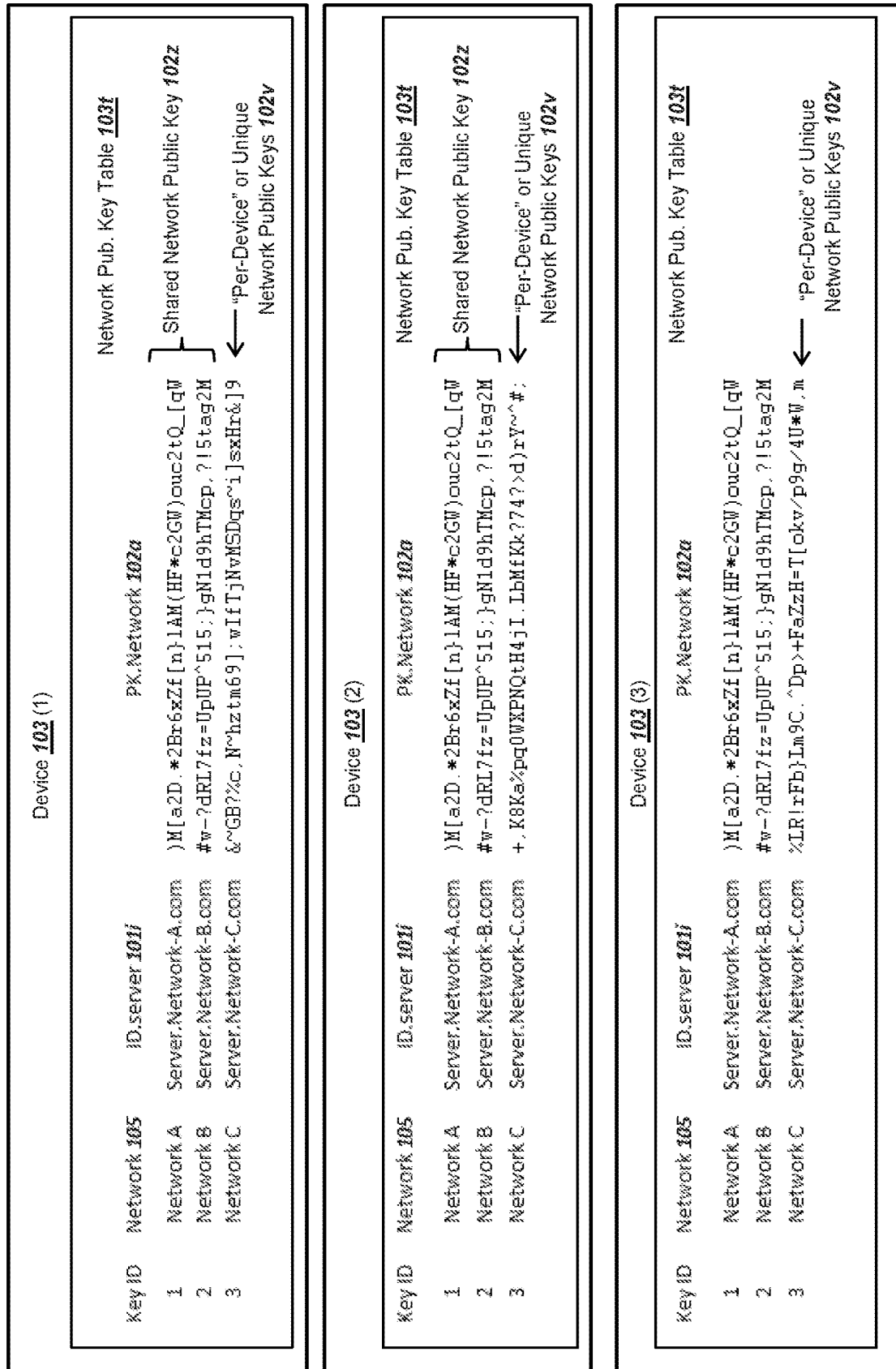


Figure 1c



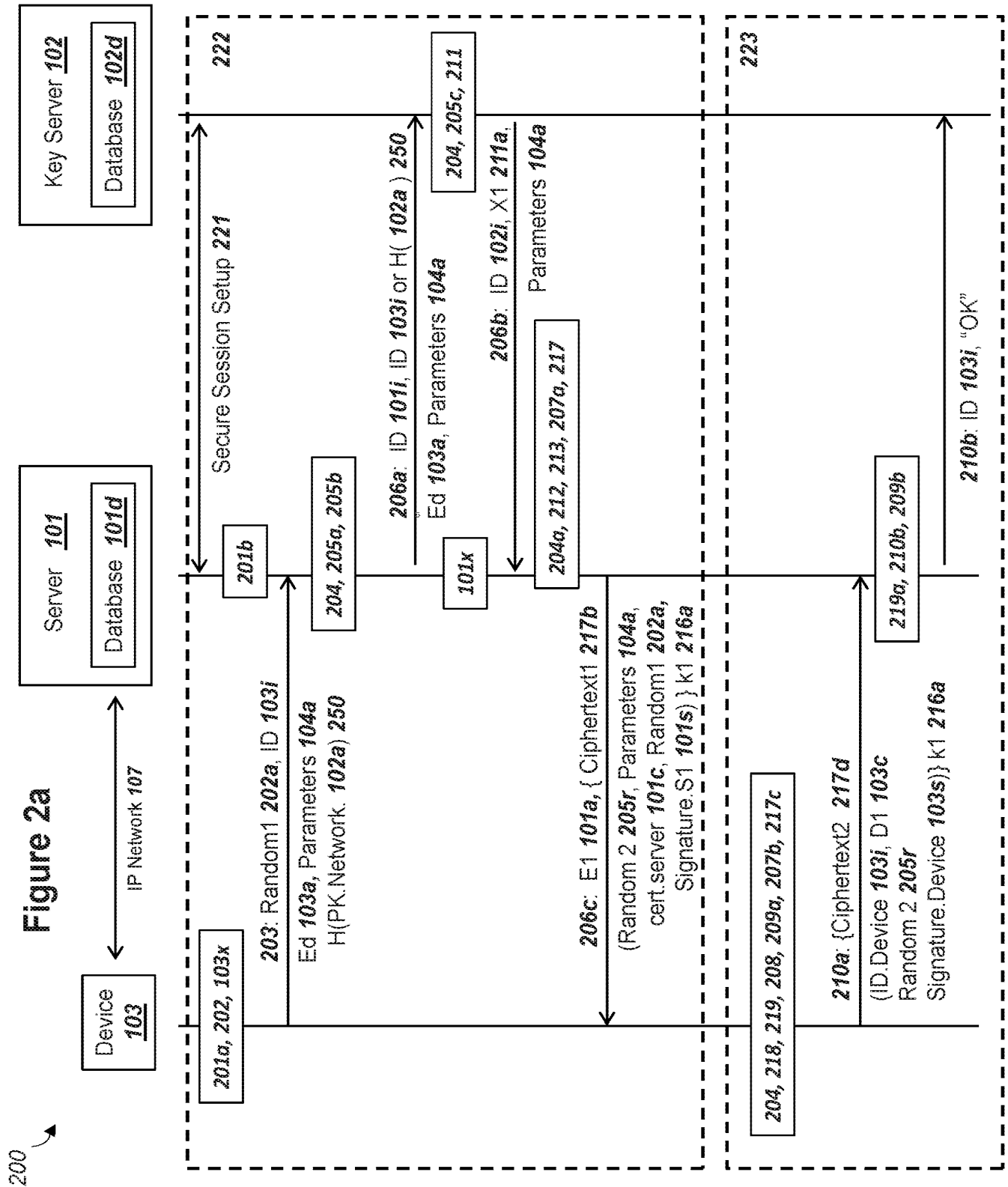


Figure 2b

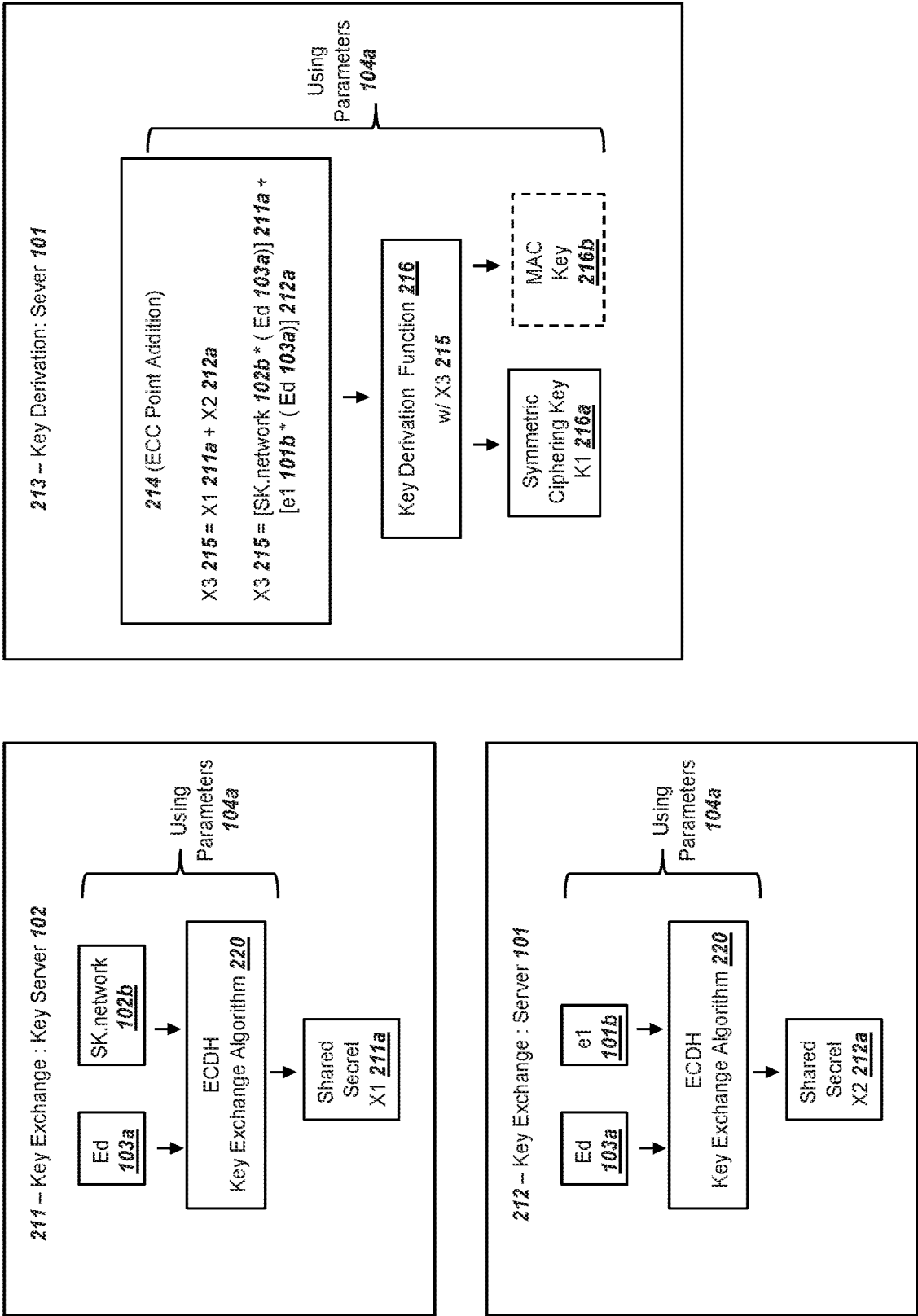


Figure 2c

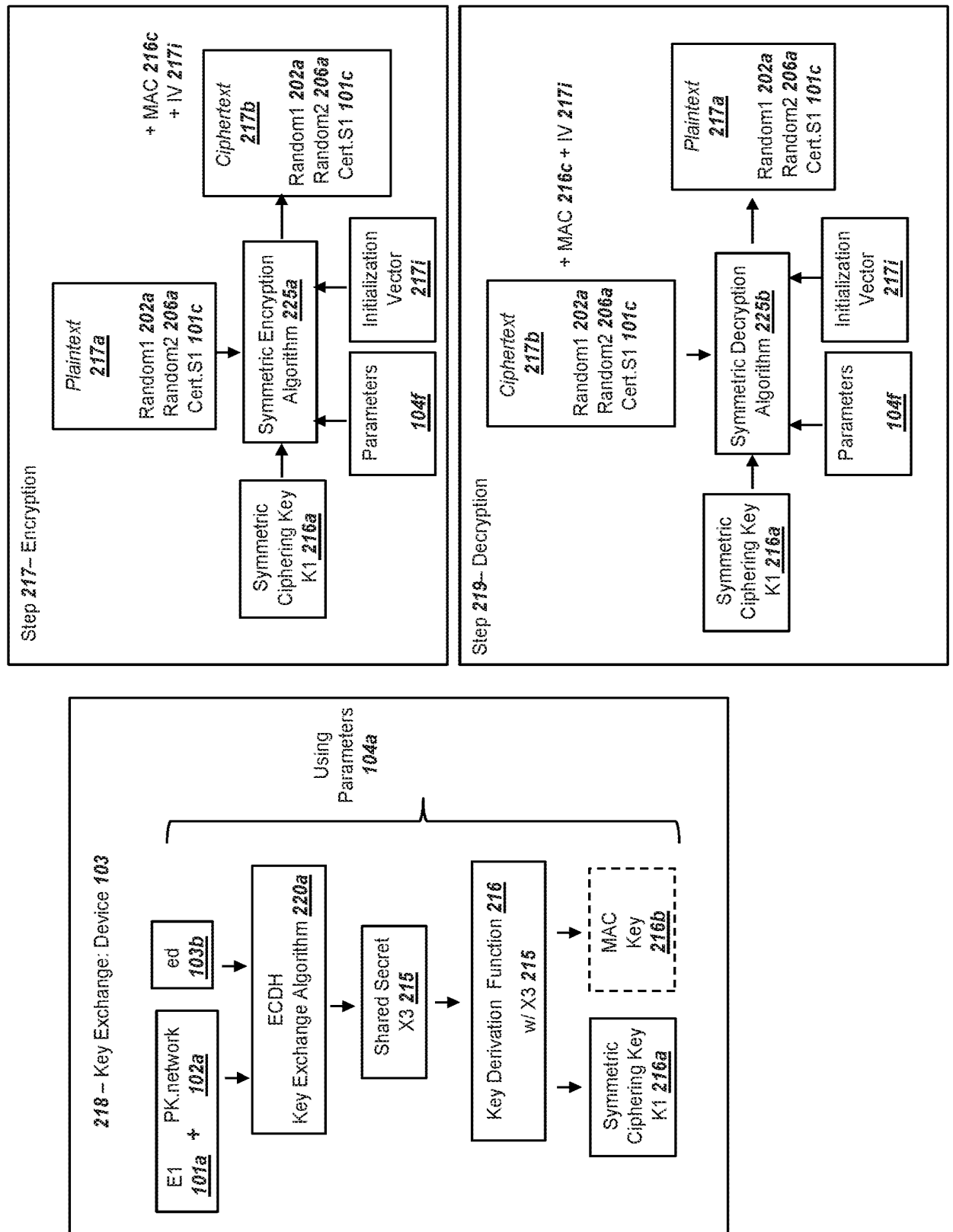


Figure 2d

Server Database 101d										
Random 1	Ed	Parameters	Source IP:Port	H(PK.network ID:key-server 102a)	X1	E1	e1	X2	X3	K1
202a	103a	104a	203a	250	102i	211a	101a	212a	215	216a
202a-1	103a-1	104a-1	203a-1	250-1	102i-1	211a-1	101a-1	212a-1	215-1	216a-1
202a-2	103a-2	104a-3	203a-2	250-2	102i-2	211a-2	101a-2	212a-2	215-2	216a-2
202a-3	103a-3	104a-1	203a-3	250-2	102i-2	211a-3	101a-3	212a-3	215-3	216a-3
● ● ●										
Cryptographic Parameters 104										
Set 104a	PKI Key		Symmetric		Symmetric		Random			
	Length	Curve	Hash	Key Length	Ciphering	Ciphering	Length	Length	Length	Length
104b	104c	104d	104e	104f	104g	104h	104i	104j	104k	104l
A	256	p256	SHA-256	256	C1	C1	128	128	128	128
B	384	p384	SHA-384	384	C2	C2	192	192	192	192
C	512	secp521r1	SHA-512	512	C3	C3	256	256	256	256
D	384	Curve25519	SHA-3	384	C4	C4	192	192	192	192
● ● ●										

Figure 2e

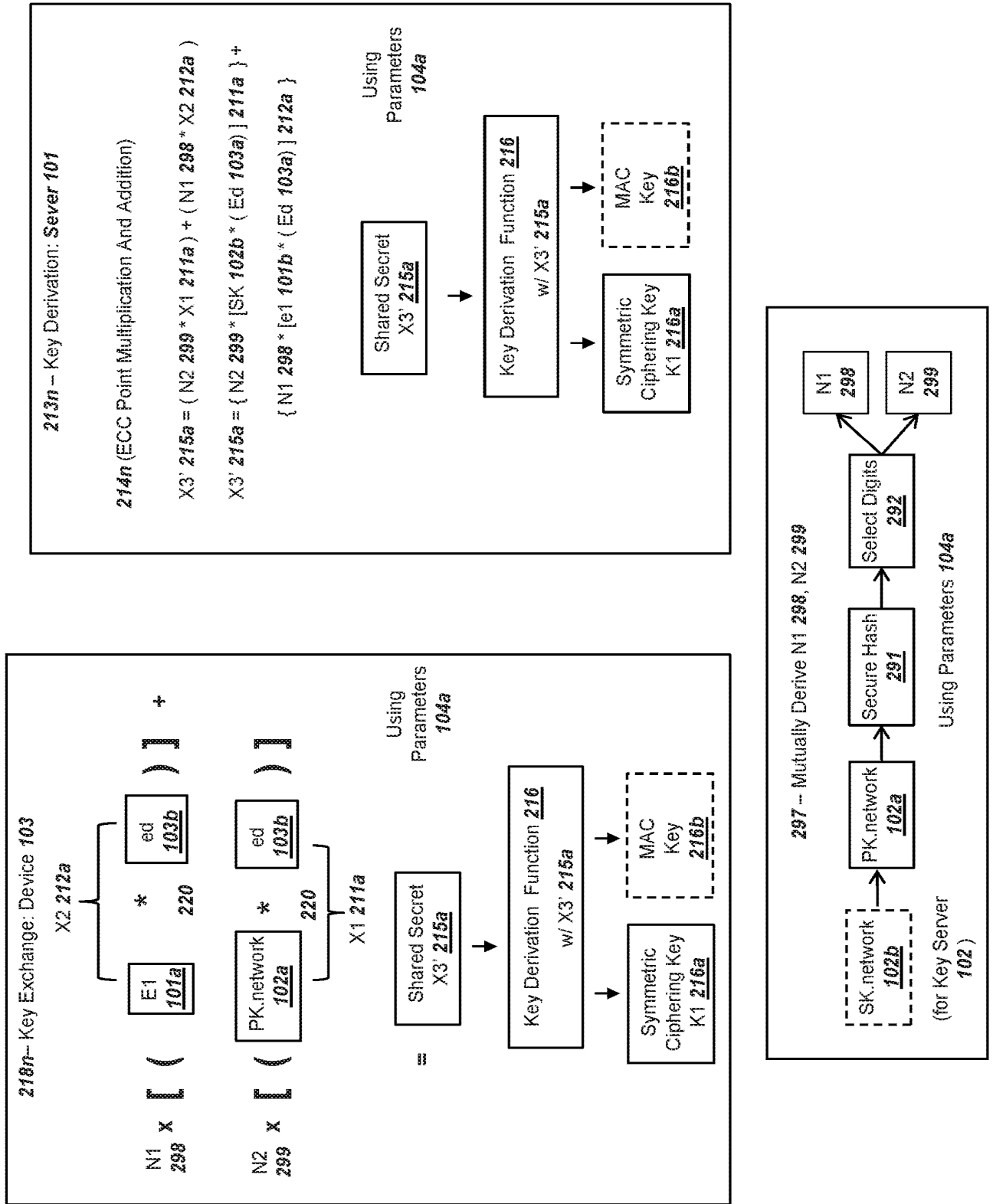


Figure 3a

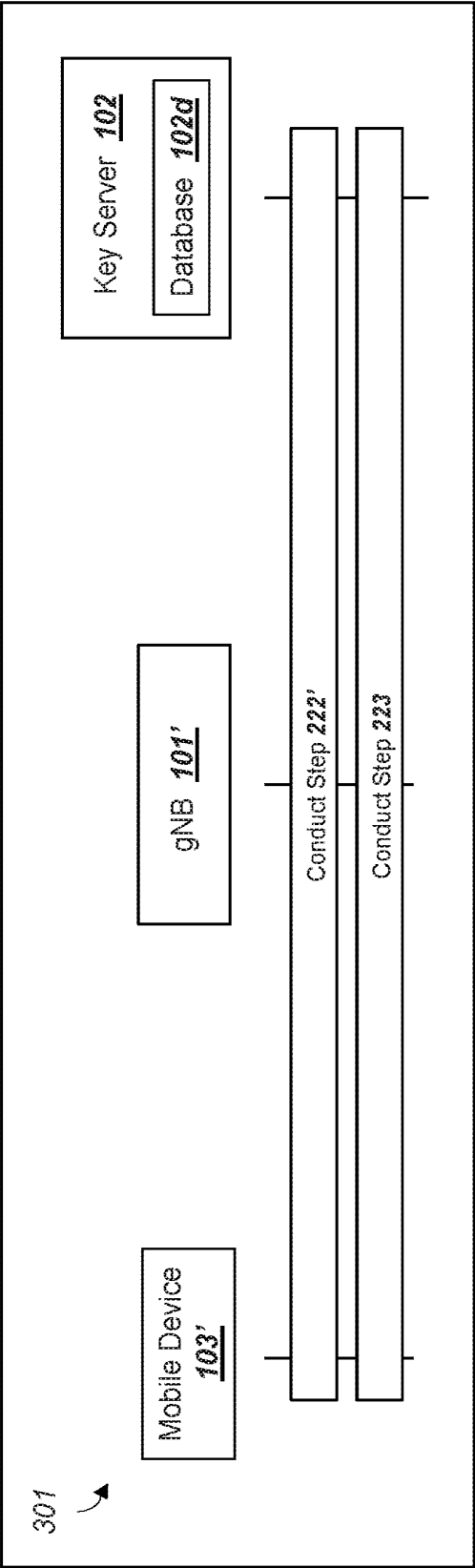


Figure 3b

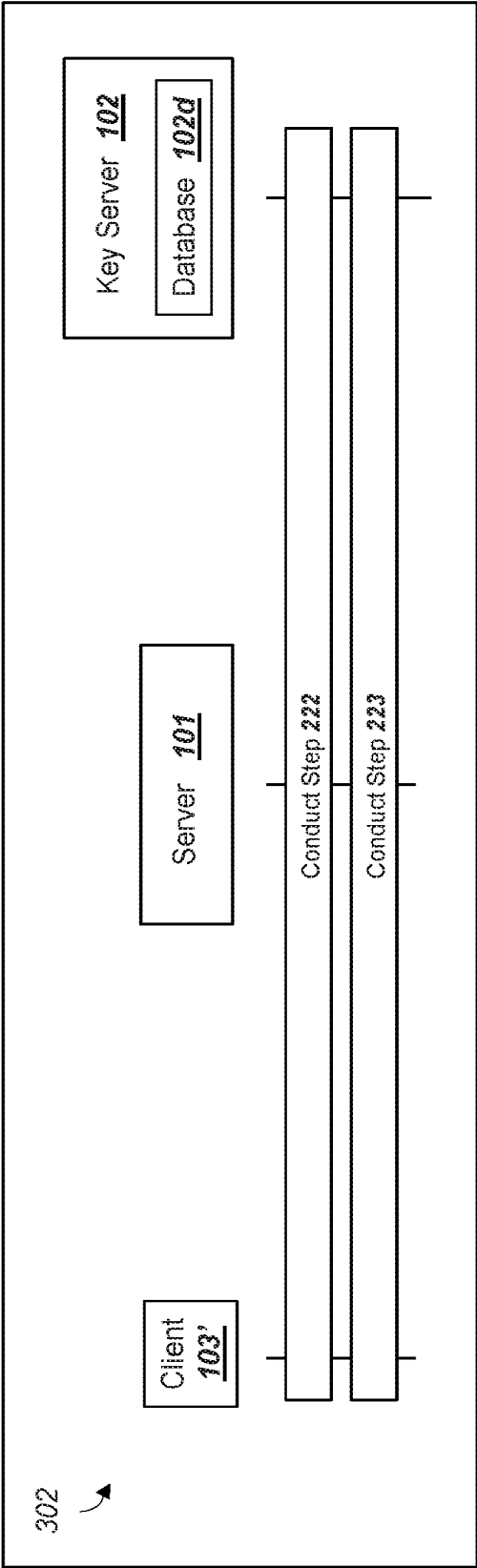
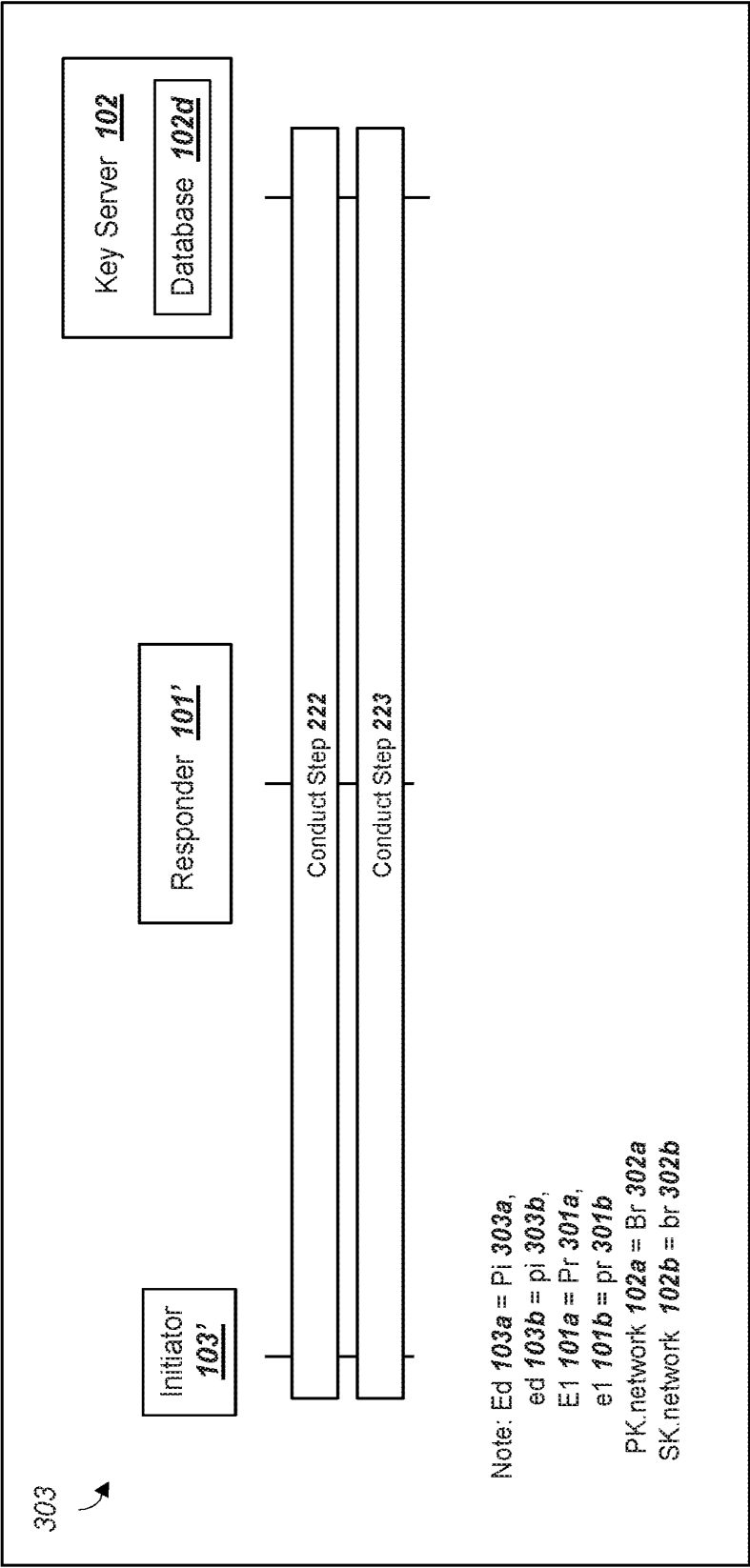


Figure 3c



INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 2019/037911

A. CLASSIFICATION OF SUBJECT MATTER

H04L 9/00 (2006.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L H04W

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

PatSearch (RUPTO internal), USPTO, PAJ, K-PION, Esp@cenet, Information Retrieval System of FIPS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2017/0237571 A1 (CLOUDFLARE, INC.) 17.08.2017, paragraphs [0043]-[0050], [0076], [0086], [0087], [0139]-[0144], [0370]-[0374]	14
A		1-6, 15-18
X	US 2015/0067338 A1 (AKAMAL YECHNOLOGIES, INC.) 05.03.2015, paragraphs [0045]-[0056]	7
A		8-10, 13
A	US 2018/0026784 A1 (MASTERCARD INTERNATIONAL INCORPORATED) 25.01.2018	1-10, 13-18



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“E” earlier document but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search

16 September 2019 (16.09.2019)

Date of mailing of the international search report

26 September 2019 (26.09.2019)

Name and mailing address of the ISA/RU:

Federal Institute of Industrial Property,
Berezhkovskaya nab., 30-1, Moscow, G-59,
GSP-3, Russia, 125993
Facsimile No: (8-495) 531-63-18, (8-499) 243-33-37

Authorized officer

P. Skulakov

Telephone No. (495) 531-65-15

Form PCT/ISA/210 (second sheet) (January 2015)

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 2019/037911

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:
2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
3. ☒ Claims Nos.: 11-12, 19-20
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- ☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- ☐ No protest accompanied the payment of additional search fees.



(51) International Patent Classification:
H04W 12/06 (2009.01) *H04W 12/04* (2009.01)

(21) International Application Number:
PCT/US2020/030424

(22) International Filing Date:
29 April 2020 (29.04.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/841,435 01 May 2019 (01.05.2019) US

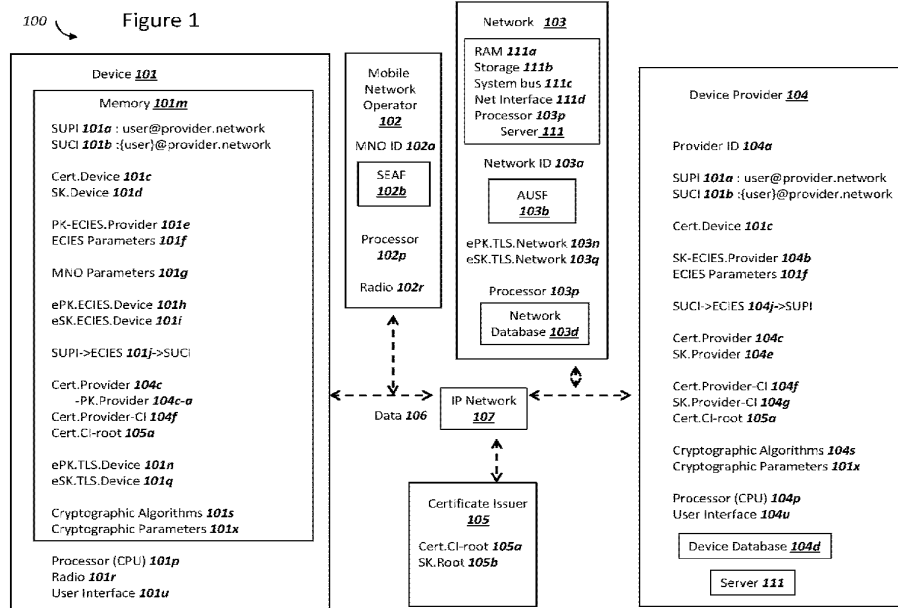
(72) Inventor; and
(71) Applicant: NIX, John, A. [US/US]; 807 Davis St., Unit
2207, Evanston, IL 60201 (US).

(74) Agent: THYMIAN, Marcus, J.; McDonnell Boehnen Hulbert & Berghoff LLP, 300 South Wacker Drive, Chicago, IL 60606 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ,

(54) Title: DISTRIBUTED EAP-TLS AUTHENTICATION FOR WIRELESS NETWORKS WITH CONCEALED SUBSCRIBER IDENTITIES



(57) Abstract: A device, mobile operator, network, and a device provider can exchange messages for EAP-TLS authentication. The network can include an authentication server function (AUSF). A device and device provider can record both a device certificate and a device provider certificate. The device provider can receive a subscriber concealed identity (SUCI) for the device from the AUSF. The device provider can decrypt the SUCI and send the network the certificates with a device SUPI. The network can send at least a TLS ephemeral public key to the device provider. The device provider can generate a digital signature for at least the ephemeral public key with a private key for the device provider certificate. The AUSF can complete an EAP-TLS authentication with the device using at least (i) the device certificate for the device, (ii) the device provider certificate for the server, (iii) the digital signature, and (iv) the ephemeral public key.

[Continued on next page]

UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

DISTRIBUTED EAP-TLS AUTHENTICATION FOR WIRELESS NETWORKS
WITH CONCEALED SUBSCRIBER IDENTITIES

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 62/841,435, which is hereby incorporated herein by reference in its entirety.

BACKGROUND

Technical Field

The present systems and methods relate to conducting an extensible authentication protocol -- transport layer security (EAP-TLS) authentication with concealed subscriber identities for wireless networks, and more particularly to storing and using keys and certificates with a device, a network, and a device provider in order to improve efficiency, increase flexibility, and enhance security of the EAP-TLS authentication.

Description of Related Art

The use of EAP-TLS authentication has been widely adopted over the past decade within enterprise wireless networks using WiFi standards such as 802.11. Standards for EAP-TLS authentication include the Internet Engineering Task Force (IETF) Request for Comments (RFC) 5216 titled "The EAP-TLS Authentication Protocol", which is herein incorporated by reference in its entirety. Over the past two years, the European Technical Standards Institute (ETSI) has begun publishing standards for 5th generation (5G) wireless wide-area networks (WAN), such as described in TS 133 501 V15.1.0 titled "5G; Security architecture and procedures for 5G System", which is herein incorporated by reference in its entirety. Unlike prior 4th generation wireless WAN networks, 5G standard networks can include the support of authentication of devices using EAP-TLS, as described in Annex B of TS 133 501. EAP-TLS authentication enables the use of certificates for mutual authentication of a device and a wireless network. Previous wireless WAN standards for authentication such as with 4G LTE, 3G, etc. have depended on a pre-shared secret key recorded in a device and a secure server within the network to conduct an authentication and key agreement (AKA) protocol. The significant changes with 5G networks removes some previous limitations, but also creates additional needs in the art in order to efficiently support EAP-TLS authentication for thousands or more different types of devices connecting with hundreds or more different wireless networks globally.

The inclusion of support for EAP-TLS in 5G networks can ease the support for wireless device authentication and secure connectivity for many applications, where a pre-shared secret key such as a key K within a subscriber identity module (SIM) or embedded SIM may no longer be required. Most commercial arrangements for device owners or device users bind the use of the pre-shared secret key K within a device to a particular home wireless network that also records and derives keys from the same

pre-shared secret key. 5G networks also include the use of subscribed concealed identities as a partially encrypted form of a subscriber permanent identity. 5G networks as specified in TS 133 501 also include the use of a network access identifier (NAI) according to IETF RFC 7542 titled “The Network Access Identifier”, which is herein incorporated by reference in its entirety. In addition, in 2018 the IETF updated TLS version 1.2 to TLS version 1.3 in IETF RFC 8446 titled “The Transport Layer Security (TLS) Protocol Version 1.3”, which is herein incorporated by reference in its entirety. ETSI published standards as of May 2019 such as TS 133 501 for 5G networks do not address the use of TLS version 1.3 with EAP-TLS authentication. A need exists in the art for devices, wireless networks, networks, and device providers to support an EAP-TLS authentication using TLS version 1.3 with the encrypted or partially encrypted subscriber concealed identities as described in TS 133 501 Annex B, where the corresponding subscriber permanent identities can use the NAI format.

Although the use of PKI certificates and private keys for EAP-TLS authentication can avoid the logistical and provisioning complexities of using a pre-shared secret key for authentication (such as with 4G and 5G AKA authentication), the use of EAP-TLS authentication can create a new series of problems or needs in the art for the many different devices and wireless networks which use EAP-TLS authentication. These problems may be avoided or more easily addressed for existing implementations of EAP-TLS authentication with corporate or enterprise WiFi networks, for example, because the enterprise can control both the device and the wireless network. The enterprise can ensure (i) a root certificate or certificate issuer certificate stored in the wireless network can be used to authenticate the device and (ii) and a root certificate or certificate issuer certificate stored in the device can be used to authenticate the wireless network. Both (i) and (ii) in the previous sentence are often required for mutual authentication using EAP-TLS standards. In addition, TLS authentication and PKI certificates and secret keys can support many different sets of cryptographic algorithms and cryptographic parameters. An enterprise deploying EAP-TLS authentication currently can control the device certificates and the wireless network certificates before device deployment and installation to ensure the PKI certificates support compatible sets of cryptographic parameters and algorithms.

However, for widespread and even global deployments of 5G networks and devices, there can be hundreds or more different device manufacturers, device providers, and wireless networks, where each could use many different cryptographic algorithms, parameters, and/or certificate issuers. In addition, (A) the device manufacturers or device providers storing device default PKI certificates in devices using device default cryptographic algorithms and cryptographic parameters may not support (B) the network PKI certificates and network supported cryptographic algorithms and cryptographic parameters. For many applications, no pre-existing relationship may exist between a device manufacturer and a wireless network before device distribution to end users and the device's first connection to a 5G wireless network. A device, using EAP-TLS authentication and a subscriber concealed identity, may not record a certificate chain or root certificate that can be used to authenticate

a network certificate received from the network in a TLS handshake for EAP-TLS authentication. A need exists in the art for a wireless network to support EAP-TLS authentication for any one of hundreds or more of different device types using concealed identities from hundreds or more different device providers. A need exists in the art, where the device uses a subscriber concealed identity, to support verification of a certificate received from any one of hundreds or more different wireless networks, where no pre-existing relationship may exist between the device owner and the wireless network.

For example with the hundreds or more different 5G wireless networks globally and with a manufactured or installed set of certificate issuer certificates or root certificates, a device could receive a server certificate from an authentication server function (AUSF) that does not chain or may not be able to be verified through a root certificate or certificate issuer certificate recorded by the device. In addition, the server certificate may use cryptographic algorithms and parameters that are different than those used by a certificate issuer certificate stored within the device. The device may not have access to an IP network without first authenticating with a wireless network. This potentially creates significant problems, where the device does not have access to the IP network to receive (i) a supporting certificate for verifying the server certificate from (ii) a server other than through the wireless network where a connection is attempted. In other words, the supporting certificate may be required in order to verify the server certificate received from the wireless network, but the device may not be able to receive the supporting certificate from an IP network without first successfully completing an EAP-TLS authentication.

Similarly, (i) the wireless network may control or establish a first certificate chain for a server certificate used by an authentication server function (AUSF), (ii) a device provider may control or establish a second certificate chain within a device before distribution to and end user for verifying the server certificate, and (iii) the server certificate from a wireless network may not be verifiable by the second certificate chain. The problem of (iii) is compounded by the thousands or more potential combinations of different certificate issuer certificates and supported cryptographic parameters for required certificates to successfully complete an EAP-TLS authentication. A need exists in the art for ensuring a device can authenticate a server certificate from a wireless network using both (i) a certificate issuer certificate recorded in the device before distribution, and (ii) cryptographic algorithms and parameters recorded in the device before distribution that would support verification of the server certificate. A need exists in the art for a device to support EAP-TLS mutual authentication, without requiring a device user to load a server certificate or supporting certificate issuer certificate in order for the device to conduct mutual authentication with a wireless network. A need exists in the art for a device to record compatible cryptographic algorithms and cryptographic parameters for a server certificate without requiring a device user to load compatible cryptographic algorithms and cryptographic parameters that can be required in order for the device to use or verify the server certificate.

Without access to an IP network (such as after successfully completing EAP-TLS authentication with the wireless network), the device may not be able to request a server certificate that can be verifiable through an installed certificate chain in a device, and consequently mutual authentication using methods such as EAP-TLS may fail or require manual intervention or unnecessary interaction with a device user in order for the EAP-TLS authentication to succeed. Again, the above problems related to server certificate verification by a device and the use of different cryptographic algorithms and parameters for different server certificates received by a device is compounded by the fact that there could globally be hundreds of mobile network operators and thousands or more device providers.

A device provider may not know which of the hundreds of different mobile networks a particular device may try to connect with until the device is turned on by a user or first attempts to connect with a mobile network in a particular geographical region. Even if a particular geographical region is known, such as the United States, before a device attempts to connect to one of a handful of wireless networks, the preferred wireless network may not be known to a device provider before a device attempts to connect. A need exists in the art for a device to receive a server certificate in a TLS handshake from a wireless network supporting EAP-TLS, where verification of the server certificate does not require intervention from an end user, in order to allow a device to mutually authenticate with EAP-TLS using a subscriber concealed identity with a plurality of different mobile network operators.

In addition, a device provider may control or establish a certificate chain for verifying a device certificate stored within a device, and (ii) an authentication server function may record a set of certificate issuer certificates and/or root certificates with cryptographic algorithms and cryptographic parameters that may not support the verification of the device certificate. Devices from hundreds or more different device providers can also use different certificate chains for the device certificates and also different sets of cryptographic algorithms and cryptographic parameters. The request by the authentication server function for supporting certificate issuer certificates and/or cryptographic algorithms and cryptographic parameters for a received device certificate may take additional time, complexity, and potential security issues for securely receiving a supporting certificate issuer certificate and/or cryptographic algorithms and cryptographic parameters in order to properly verify a received device certificate. A need exists in the art for ensuring an authentication server function can securely and efficiently authenticate a device certificate received through a wireless network, where the device can utilize a subscriber concealed identity.

Many other examples exist as well for needs in the art for devices to securely and efficiently conduct EAP-TLS authentication with wireless networks, where the device uses a subscriber concealed identity, and the above are examples are just a few and intended to be illustrative instead of limiting.

SUMMARY

Methods and systems are provided for a device, a mobile network operator (MNO), a network, and a device provider to support EAP-TLS authentication, where the device can use a subscriber concealed identity. A system can include a plurality of devices, a mobile network operator, a network, and a device provider. The device can record a subscriber permanent identifier (SUPI), a device certificate, a device private key corresponding to the device public key in the certificate, an elliptic curve integrated encryption scheme (ECIES) function, a static provider public key for the ECIES function, MNO parameters, a device provider certificate, and cryptographic algorithms and cryptographic parameters used with both the device provider certificate and the device certificate. The subscriber permanent identifier can comprise a network access identifier (NAI) supporting IETF RFC 7542, where the NAI can include a user name and a realm. The realm can comprise a first portion with a provider identity and a second portion with a network identity.

The device can be a computing device for connecting with wireless networks such as a 5G network or a Wi-Fi network and the device can include a radio. The device can include a processor, memory, a user interface, and a bus connecting the components within an enclosure. The device could comprise a radio module within a larger device, such as a radio module within an automobile, industrial equipment, a mobile handset such as a smart phone, and other possibilities exist as well for physical embodiments of the device. The device can conduct an EAP-TLS authentication through the MNO with the network.

The mobile network operator can comprise a wireless network with geographically distributed radios and antennas in order to wirelessly connect with the plurality of devices. The MNO can include a MNO identity in order to uniquely identify the MNO, such as including a mobile country code and a mobile network code. The MNO can receive messages from the device through a wireless network and forward the messages to the network. The MNO can receive messages from the network and forward the messages to the device through the wireless network. The MNO can establish a secure session through an IP network with the network.

The network can include a plurality of servers operating in a coordinate manner to support communications with a device. The plurality of servers for the network can comprise a server system. The network can include an authentication server function (AUSF) and a network database. The network can include an identity comprising the network identity in the second portion of a realm for the device subscriber permanent identifier. The network can conduct an EAP-TLS authentication with the device through the MNO supporting 5G standard such as TS 133 501 V15.1.0. The network can establish a secure session with a device provider. The network can operate a network database for storing data associated with each of the plurality of different devices connecting with the network through the MNO.

The device provider can comprise a collection of servers for an entity that controls the device before the device connects with the MNO. The device provider can represent a device manufacturer, a device distributor, or a device owner for a plurality of different devices. The device provider can include an identity comprising the provider identity in the first portion of a realm for the device subscriber permanent identifier. The device provider can operate a device database storing data for the plurality of devices. The device provider can record the subscriber permanent identifier, a device certificate for the device, an elliptic curve integrated encryption scheme (ECIES) function for decryption of a subscriber concealed identity (SUCI), a static provider private key for the ECIES function, a device provider certificate, a device provider private key for the public key in the device provider certificate, and cryptographic algorithms and cryptographic parameters used with at least the device provider certificate.

The device can generate the subscriber concealed identity (SUCI) using at least the SUPI and the ECIES encryption function. The device can use the MNO parameters to establish a radio resource connection with the MNO. The device can send the SUCI to the MNO, and the MNO can use the plaintext second portion of the realm to select the network and send the SUCI to the network. The network can use the plaintext first portion of the realm to select the device provider and send the SUCI to the device provider. The device provider can use at least the ECIES function and the static provider private key for the ECIES function to decrypt the SUCI and read the SUPI. The device provider can use the SUPI to select a device provider certificate and a device certificate for the device from the device database. The device provider can send at least the device provider certificate and the device certificate to the network, along with the SUCI. The network can store the received device provider certificate and the device certificate along with the SUCI in the network database.

The network can send an EAP-TLS start command to the device through the MNO. The network can receive a "client hello" and the SUCI from the MNO, where the "client hello" can support TLS version 1.3 as specified in RFC 8446. The "client hello" can include an ephemeral elliptic curve cryptography (ECC) public key for the device, where the device stores the corresponding device ephemeral private key. The AUSF can generate a server ephemeral PKI key pair compatible with the received ECC public key from the device in the "client hello" message, including a server ephemeral public key. The network can store the PKI keys in the network database. The network can use the SUCI to select the device provider certificate received from the device provider in the paragraph above as a server certificate for conducting the TLS version 1.3 handshake. The network can use the SUCI received with the "client hello" to select the device provider certificate from the network database. The network can process a "server hello" using at least the selected device provider certificate. Data processed by the AUSF for a "server hello" can include standard data such as the server ephemeral public key, selected extensions, a certificate request, and the selected device provider certificate. The data processed in the previous sentence can be sent in a "message to sign" request to the device provider

from the network. The device provider can use the device provider secret key (which can correspond to the device provider public key in the device provider certificate) to digitally sign the message to sign using a digital signature algorithm and parameters specified in the device provider certificate. The resulting digital signature can comprise a "CertificateVerify" portion of a "server hello" message. The device provider can send the resulting digital signature to the network.

The network can generate a shared secret and a symmetric ciphering key using the device ephemeral public key received in the "client hello" and the derived server ephemeral private key with an elliptic curve Diffie-Hellman (ECDH) key exchange. The network can store the derived shared secret and the symmetric ciphering key with the SUCI for the device in the network database. The server can encrypt at least the selected extensions, a certificate request, and the selected device provider certificate and the "CertificateVerify" into a ciphertext using the derived symmetric ciphering key. The network can send the device a "server hello" message with the server ephemeral public key and the ciphertext through the MNO, where the message to the MNO can include the SUCI so that the MNO can route the message to the device.

The device can receive the "server hello" message and take steps to process the message. The device can generate the shared secret and the symmetric ciphering key using the server ephemeral public key received in the "server hello" and the derived device ephemeral private key using the ECDH key exchange. The device can decrypt the ciphertext using the symmetric ciphering key. The device can read the plaintext value for the device provider certificate. The device can verify that the received device provider certificate is the same as the device provider certificate recorded by the device before the device connected with the MNO. In this manner, the device provider certificate as used by the AUSF and MNO can be trusted without requiring the device to verify the received device provider certificate through a stored certificate issuer certificate. If the received device provider certificate is not the same as a recorded device provider certificate, then the device could verify the received device provider certificate through a certificate issuer chain stored within the device.

In other words, the device does not need Internet connectivity in order to verify the received device provider certificate as a server certificate in the "server hello" message because the device provider certificate can be verified with data previously stored inside the device before connecting with the MNO. The selection of the received device provider certificate above by the device provider (by sending the device provider certificate to the network) can ensure the device provider certificate can be verified by the device without the device needing connectivity to an IP network (in order to query for other intermediate verifying certificates), thereby solving some needs in the art identified above.

The device can also verify the "CertificateVerify" digital signature portion of the decrypted "server hello" ciphertext using the public key with the received device provider certificate. In this manner, the device can trust that the AUSF has access to the device provider secret key corresponding to the public key in the device provider certificate (although the device provider and not the AUSF may

store and operate with the device provider secret key). Verification of the “CertificateVerify” digital signature can comprise an authentication of the network and/or MNO with the device.

The device can use the device provider certificate and the user portion of the SUPI to select a device certificate stored in the device. For example, the device could record multiple device certificates and select a device certificate compatible with the cryptographic parameters and algorithms within the device provider certificate. Or the device could select the device certificate based on the extensions within the decrypted ciphertext within the “server hello” message. The device can generate a response to the “server hello”, where the response includes the selected device certificate and a digital signature or “CertificateVerify” from the device. The device could generate the “CertificateVerify” using the device private key corresponding to the device public key in the device certificate selected by the device. The device can encrypt the response using at least the mutually derived shared secret and data from the “client hello” and “server hello” messages. The device can send the encrypted data in the response to the MNO. The MNO can forward the response with the SUCI to the network.

The network/AUSF can use the SUCI received with the encrypted response to select the mutually derived shared secret and/or the symmetric ciphering key from the network database. The network/AUSF can decrypt the encrypted response from the device using at least the mutually derived shared secret to read the plaintext values of the device certificate and the digital signature for the device “CertificateVerify”. The AUSF can compare (i) the received device certificate from the device with (ii) the device certificate stored in the network database, where the stored device certificate in the network database could be received previously as described above from the device provider. If (i) the received device certificate from the device and (ii) the received device certificate from the device provider are the same, then the device certificate can be trusted or considered verified. Or, if the device certificate received from the device is not the same as the device certificate received from the device provider, then the AUSF can conduct a certificate verification for the device certificate received from the device through a chain of certificate issuer certificates. The AUSF can verify the device “CertificateVerify” in the response by generating the equivalent of a digital signature or a hash value using the device public key from the device certificate, and if the received digital signature is equal to or the same as the generated digital signature, then the device can be considered authenticated with the network.

The AUSF can use (i) the shared secret key from the TLS ECDH key exchange stored in the network database for the device and (ii) the plaintext data from the “client hello” and the “server hello” in order to generate a TLS master secret according to the steps in IETF RFC 8446, such as section 7.1 for the key schedule. The AUSF can use the generated or derived TLS master secret to generate at least a value for an extended master secret key (EMSK) as specified in EAP-TLS standards, such as inputting the TLS master secret into a key derivation function that could include a secure hash algorithm. The AUSF can use a portion of the EMSK, such as the leading 256 bits from an EMSK as an anchor key or Kausf. The AUSF can send the anchor key along with the SUPI for the device to the MNO. The MNO

can use the anchor key to derive subsequent keys required for secure wireless communications with the device.

The device can conduct the equivalent steps of using the shared secret key and the plaintext data from the “client hello” and the “server hello” in order to generate the TLS master secret according to the steps in IETF RFC 8446. The device can use the generated or derived TLS master secret to generate at least a value for EMSK as specified in EAP-TLS standards, such as inputting the TLS master secret into a key derivation function that could include a secure hash algorithm. The device can use a portion of the EMSK, such as the leading 256 bits from an EMSK as an anchor key or Kausf. The device can use the anchor key to derive subsequent keys required for secure wireless communications with the mobile network operator.

These as well as other aspects and advantages will become apparent to those of ordinary skill in the art by reading the following detailed description, with reference where appropriate to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Various exemplary embodiments are described herein with reference to the following drawings, wherein like numerals denote like entities.

Figure 1 is a graphical illustration of an exemplary system, where a device communicates data with a mobile network operator and a network in order to mutually authenticate with the network, in accordance with exemplary embodiments;

Figure 2 is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device, a mobile network operator, a network, and a device provider, in accordance with exemplary embodiments; and

Figure 3 is an illustration of a device database and a network database with exemplary data stored, in accordance with exemplary embodiments.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS**Figure 1**

Figure 1 is a graphical illustration of an exemplary system, where a device communicates data with a mobile network operator and a network in order to mutually authenticate with the network, in accordance with exemplary embodiments. The system 100 can include a device 101, a mobile network operator 102, a network 103, a device provider 104, and a certificate issuer 105. The depicted nodes or entities can communicate data 106 over an Internet Protocol (IP) network 107. Although a single device 101, a single mobile network operator 102, a single network 103, a single device provider 104, and a single certificate issuer 105 are depicted in Figure 1, a system 100 can comprise a plurality of each of the depicted nodes or entities. A system 100 as depicted in Figure 1 can support EAP-TLS authentication for 5G wireless WAN networks, such as supporting EAP-TLS authentication as described in Annex B of TS 133 501 V15.1.0, although other wireless networking technology for device 101 and MNO 102 could be supported as well.

Device 101 and mobile network operator 102 can utilize a variety of wireless wide area network (WAN) and wireless local area network (LAN) wireless and technologies to communicate data 106 between the nodes, including Low Power Wide Area (LPWA) technology, 3rd Generation Partnership Project (3GPP) technology such as, but not limited to, 3G, 4G Long-Term Evolution (LTE), or 4G LTE Advanced, NarrowBand – Internet of Things (NB-IoT), LTE Cat M, and 5G or subsequent wireless technologies. In addition, the wireless technology used by device 101 and mobile network operator 102 could support or implement wireless LAN technologies such as WiFi and the related series of standards from IEEE 802.11 standards, such as 802.11ac, 802.11ax, etc. Other examples exist as well for wireless WAN technology and/or wireless LAN technology used by device 101 and mobile network operator 102 without departing from the scope of the present disclosure.

Mobile network operator 102, network 103, and device provider 104 can connect to the IP network 107 and communicate with each other via a wired connection such as, but not limited to, an Ethernet connection, or a fiber optic connection. In other words, mobile network operator 102 can connect to (i) device 101 through wireless technology and (ii) a network 103, a device provider 104, and a certificate issuer 105 using wired technology. IP network 107 could also be a public or private network supporting Internet Engineering Task Force (IETF) standards such as, but not limited to, such as, RFC 786 (User Datagram Protocol), RFC 793 (Transmission Control Protocol), and related protocols including IPv6 or IPv4. A public IP network 107 could utilize globally routable IP addresses. A private IP network overlayed on IP network 107 could utilize private IP addresses which could also be referred to as an Intranet. Other possibilities for IP Network 107 and a private network between mobile network operator 102, network 103, and device provider 104 exist as well without departing from the scope of the disclosure.

Mobile network operator (MNO) 102 can include a plurality of radio-frequency (RF) access technologies (RAT) systems for supporting wireless communications with a plurality of devices 101 in a networked manner, including through the use of geographically dispersed antennas, radio nodes, or towers. The RAT systems for MNO 102 can include a radio 102r for communicating with device 101, where radio 102r includes an antenna system and can operate through licensed radio spectrum. In exemplary embodiments, MNO 102 can operate a plurality of radios 102r which are connected to an IP network 107 in a secure manner, including connecting via a private IP network.

In exemplary embodiments, mobile network operator 102 can comprise traditional public land mobile networks providing data and voice services such as AT&T, Verizon, T-Mobile, Sprint, etc, and provide data communications services through a variety of radio access technologies. Further, although Figure 1 depicts an MNO 102 as communicating with device 101 with a radio 102r, another entity besides a mobile network operator could perform the function of MNO 102. For embodiments where radio 101r for device 101 uses WiFi technology, then MNO 102 could comprise an entity other than a PLMN, such as a service or network that operates a plurality of WiFi access points. The function of MNO 102 could also be conducted by a large enterprise with a collection of geographically distributed WiFi access points.

Each of mobile network operator (MNO) 102, network 103, and device provider 104 could operate a plurality of servers in order to support the communications and connectivity depicted in Figure 1 and also Figure 2 below. The exemplary data structures, values or numbers, and steps for a MNO 102, network 103, and device provider 104 depicted in Figure 1 and also Figure 2 below could be recorded and/or conducted by a collection of servers for each entity. Exemplary servers for a mobile network operator 102, network 103, or device provider 104 in system 100 can be either different physical computers such as rack-mounted servers, or different logical or virtual servers or instances operating in a “cloud” configuration.

An exemplary server or collection of servers for MNO 102 can comprise a Security Anchor Function (SEAF) 102b, where a MNO 102 can include at least one SEAF 102b for establishing secure and authenticated communications with device 101 through at least one radio 102r. The SEAF 102b could comprise and operate according to the European Technical Standards Institute (ETSI) standard TS 133 501 V15.1.0 from August of 2018 and titled “5G; Security architecture and procedures for 5G System”, which is hereby incorporated by reference in its entirety. In exemplary embodiments, the SEAF 102b for MNO 102 can receive an anchor key from network 103 in order to establish secure and authenticated communication with device 101, such as depicted and described for a message 237 below in Figure 2. In exemplary embodiments where Extensible Authentication Protocol – Transport Layer Security (EAP-TLS) in a system 100, then SEAF 102b could operate as specified in section 6.1.1.2 of TS 133 501 V15.1.0, where “the SEAF takes the role of pass-through authenticator.”

Note that MNO 102 could use other wireless networking technologies besides 5G. Thus, although an SEAF 102b is depicted in Figure 1, a different collection of servers or an entity besides a SEAF 102b could perform the steps for a MNO 102 depicted in Figure 2 below, without departing from the scope of the present invention. In general, the function of an SEAF 102b as depicted and described in connection with Figure 2 below could be conducted by at least one server or process operating within a cloud computing environment, where the server or process may not be an SEAF 102b. For example, the server or process operating with equivalent functionality as an SEAF 102b in Figure 2 below could (i) receive a partially encrypted Network Access Identifier (NAI) such as a SUCI 101b from device 101 with an encrypted user name and a plaintext realm for the NAI, and (ii) use the plaintext realm to forward the SUCI 101b to network 103, where network 103 uses or is associated with at least a portion of the plaintext realm.

MNO 102 can include a MNO identity 102a, where MNO identity 102a can comprise the combination of a mobile country code and a mobile network code, according to 3GPP standards for 4G networks, 5G networks, etc. As one example, MNO identity 102a for an example network of AT&T could comprise an MCC number of 310 and a MNC number of 410, such that the MNO identity 102a can be 310410. Note that a MNO 102 can operate with multiple MNO identities 102a, and a MNO identity 102a could also comprise a Domain Name Service (DNS) name such as an exemplary value of “att.net”, and other possibilities exist for a MNO identity 102a without departing from the scope of the present invention. In addition, a MNO 102 could operate with a plurality of MNO identities 102a.

MNO 102 can include a plurality of processors 102p in order to store and record data as well as communicate within a system 100 over an IP network 107 and a radio 102r. Processor 102p can comprise a general purpose processor appropriate for the computational requirements for a MNO 102, and may operate with multiple different processor cores, including field programmable gate arrays (FPGA). A processor 102p can comprise exemplary ARM® based processors or an Intel® based processor such as belonging to the XEON® family of processors, and other possibilities exist as well. Processor 102p can utilize a data bus to fetch instructions from memory or storage within a server and operate on the instruction. A processor 102p can include components such as registers, accumulators, and logic elements to add, subtract, multiply, and divide numerical values and record the results in memory. In exemplary embodiments, at least one processor 102p within MNO 102 can be used with an SEAF 102b to conduct the steps for MNO 102 and send/receive messages for MNO 102 as depicted and described in connection with Figure 2 below. Although not depicted in Figure 1, MNO 102 can include memory and at least one database in order to establish communications with a plurality of devices 101 through a plurality of radios 102r.

Device 101 can be a computing device for sending and receiving data using a radio 101r. Device 101 can take several different embodiments, such as a general purpose personal computer, a mobile phone or mobile handset based on the Android® or Fuchsia from Google® or the IOS operating system

from Apple ®, a tablet, a device with a sensor or actuator for the “Internet of Things”, a module for “machine to machine” communications, a device that connects to a wireless Wide Area Network (WAN) operated by MNO 102, a router, and/or a server, and other possibilities exist as well for the embodiments of a device 101 without departing from the scope of the present invention.

The electrical components within device 101 can include a memory 101m, a processor 101p, a radio 101r, and a user interface 101u. Although not depicted in Figure 1, a data bus or a system bus could internally electrically connect the depicted components within a device 101. Additional components to support the operation of device 101 can include a battery to store electrical power, an antenna to transmit and receive RF signals, a sensor to collect data external or internal to the device 101, such as temperature, motion, position, pressure, etc. A device 101 could also include an actuator to convert electrical signals into physical actions, such as a motor for moving components, a relay for opening or closing a circuit, etc.

Memory 101m can comprise combinations of (i) volatile random access memory and nonvolatile memory. The volatile memory can include random access memory (RAM) for relatively fast read and write operations compared to nonvolatile memory, such as SRAM or DRAM. RAM for memory 101m could also include persistent RAM or nonvolatile RAM (NVRAM), such that data in a persistent RAM memory or nonvolatile RAM is stored when power is removed. Nonvolatile memory can include storage memory such as a flash memory and record or store data when power is removed from device 101. In general, different forms and electrical components for memory 101m can be used without departing from the scope of the present disclosure. Processor 101p can comprise a central processing unit (CPU) or a “system on a chip” and be similar to a processor 102p for a MNO 102 described above, but with reduced capabilities for a device 101 compared to a processor 102p for MNO.

Device 101 may include radio 101r support radio-frequency (RF) communications with networks including a MNO 102 via standards such as GSM, UMTS, mobile WiMax, CDMA, LTE, LTE Advanced, 5G, and/or other mobile-network technologies. In a wireless configuration, the radio 101r may also provide connectivity to local networks such as 802.11 WLAN, Bluetooth, Zigbee, or an IEEE 802.15.4 network, among other possibilities. In exemplary embodiments, a radio 101r is connected to an antenna, which could be either internal to device 101 or external to device 101.

Note that device 101 may also optionally include user interface 101u which may include one or more devices for receiving inputs and/or one or more devices for conveying outputs. User interfaces are known in the art and thus user interfaces are not described in detail here. User interface 101u could comprise a touch screen if device 101 operates as a smartphone or mobile phone. Device 101 can optionally omit a user interface 101u, since no user input may be required for many M2M applications such as networked sensors, although a user interface 101u could be included with device 101. LED lights or a display of LEDs could also comprise a user interface 101u.

Device 101 can include a device identity comprising a subscriber permanent identity (SUPI) 101a. The device identity can comprise a string or number to uniquely identify device 101 with MNO 102, network 103, and device provider 104, and other nodes connected to the IP network 107. A device identity of a SUPI 101a can include a network access identifier (NAI) according to IETF RFC 7542, titled “The Network Access Identifier”, which is herein incorporated by reference in its entirety. A NAI can consist of different portions, such as a user identity, and a realm. The user identity can comprise a number or a string to uniquely identify device 101 within a realm. A NAI for a SUPI 101a can have many forms, and with several examples for a NAI are included in RFC 7542. Other possibilities exist as well for a NAI as a SUPI 101a for an identity of device 101 without departing from the scope of the present invention. The use of a SUPI 101a in an NAI format as a device identity for device 101 is not required for some exemplary embodiments, and for some embodiments the SUPI 101a could be a single string with an embedded user identity and realm but without the “@” character standard for a NAI.

As depicted in Figure 1, in exemplary embodiments, a SUPI 101a can have a user identity and a realm, where the realm can have a prefix and a suffix. The prefix and the suffix can have multiple portions or more than one portion. In general, (i) a realm suffix can include a base string of a domain name associated with a network 103 (depicted as “network” for SUPI 101a in Figure 1), and (ii) a realm prefix can include a second string associated with a device provider 104 (depicted as “provider” for SUPI 101a). In exemplary embodiments, the realm prefix can include at least a portion of a provider identity 104a and the realm suffix can include at least a portion of a network identity 103a. Note that a realm suffix could also include data or a name for MNO 102, which could include a MNO ID 102a. Other embodiments of a SUPI 101a for a device 101 than that depicted in Figure 1 could include exemplary values of “user@network.provider” (e.g. a prefix first portion of the realm includes a name for the network 103 and a suffix second portion of the realm includes a name for the device provider 104). In other exemplary embodiments, the realm of a SUPI 101a can comprise solely a name or identity for device provider ID 104a, and for these embodiments, MNO 102 and network 103 could have agreements and technical arrangements to route data 106 with the realm of a provider ID 104a to device provider 104.

In preferred embodiments, a device 101 with a user identity of “XXXXXX” could have be associated with a device provider with the name “Company”, where the company has a commercial agreement and technical agreement with a cloud service such as Amazon Web Services to operate the authentication server function (AUSF) 103b, such that devices 101 for the company (e.g. a device provider) could roam and/or be authenticated with a plurality of mobile network operators. The cloud service could have commercial and technical agreements with a plurality of mobile network operators, including MNO 102 depicted in Figure 1. The cloud service could have an exemplary network ID 103a of aws.com. For this exemplary embodiment described in this paragraph, the SUPI 101a for device 101 could comprise an example string of “XXXXXX@company.aws.com”. Note that an identity of MNO

102 such as MNO ID 102a may not need to be included in SUPI 101a, and MNO 102 can select network 103 based on the suffix of the realm (e.g. aws.com). In another embodiment, the MNO 102 identity could be included in the realm, where the MNO ID 102a could be “att.com”, and the SUPI 101b for device 101 could be “XXXXXX@company.aws.att.com”. Other possibilities exist as well without departing from the scope of the present invention.

Although not depicted in Figure 1, a NAI as a SUPI 101a could comprise a string where realm has an identity of the network 102a in the prefix and an identity of the device provider 103a in the suffix. Or, the realm could comprise a single base string that is unique for a combination of an identity of device provider 104a and network 102a (such as a realm of “company-aws.com”). Further, in some exemplary embodiments, the realm of a NAI for SUPI 101a could omit an identity device provider 104a and consist only of an identity for network 103 of network ID 103a (where the realm could be uniquely associated with a device provider 104). Or, the realm of a NAI for SUPI 101a could omit an identity network 102a and consist only of an identity for device provider 103 of provider ID 103a. In general, the realm of a SUPI 101a can include sufficient identifying information (such as at least a portion of network ID 103a and/or provider ID 104a) in order for data within a message from device 101 to be (i) received by MNO 102 and forwarded to network 103, and (ii) received by network 103 and forwarded to device provider 104.

A device 101 can also include a subscriber concealed identity (SUCI) 101b. SUCI 101b can include an encrypted identity of a user name or user identity, depicted as “{user}” in Figure 1 and the plaintext realm, where exemplary formats or contents of a realm were depicted and described in the two paragraphs above. A SUCI 101b can be derived from a SUPI 101a using an elliptic curve integrated encryption scheme (ECIES) 101j, which is described in addition detail below. In exemplary embodiments, device 101 communicates with a MNO 102 using a SUCI 101b as the identity for device 101 instead of the SUPI 101a, in order for the user identity of device 101 to be encrypted.

Although “user” is the term depicted in Figure 1 and also for a NAI in RFC 7542 and related 3GPP specifications, the “user” portion of a NAI can comprise a string of digits or numbers or characters to uniquely identify device 101 for a device provider 104 and the “user” portion of a NAI within a SUPI 101a or SUCI 101b does not need to be associated with a user such as a person. In other words, a “user” within an identifier such as a NAI or a SUPI 101a can be an identity for a device. The plaintext for the realm in a SUCI 101b can be required in order for MNO 102 and/or network 103 to route data for a SUCI 101b forward (e.g. MNO 102 -> network 103 and network 103-> device provider 104). MNO 102 could use a first plaintext prefix with network ID 103a in the realm to select a network 102. In other words, in exemplary embodiments, the realm portion of a NAI for a SUCI 101b can remain plaintext, although the “user” portion of a NAI for a SUCI 101b can be encrypted.

The “user” portion of a NAI for a SUPI 101a could comprise a medium access control (MAC) address for a physical interface such as Ethernet or WiFi, an international mobile subscriber identity

(IMSI) or international mobile equipment identity (IMEI) with 2G/3G/4G networks, and other possibilities exist as well for the user portion of a NAI as a SUPI 101a without departing from the scope of the present invention. In exemplary embodiments, SUPI 101a can be written to hardware in device 101 and operate as a globally unique, long-term identity for device 101. Note that a device 101 could use several different SUPI 101a, such as a first SUPI 101a with a first network 103 and a second SUPI 101a with a second network 103, where the realm portion can be different for the first and second SUPI 101a and the different realm portions can include different network IDs 103a.

A certificate for device 101 can comprise cert.device 101c and can include the device identity comprising the SUPI 101a. For some embodiments, the device identity within a device certificate cert.device 101c can be a number, string, or value that is different than the SUPI 101a. As one example, a SUPI 101a could be for connecting with a wireless network 102, while a device identity in a cert.device 101c can be for communicating with a device provider 104. A certificate 101c for device 101 can include a device public key corresponding to the device private key SK.Device 101d, a set of cryptographic parameters for the device public key, the identity of a certificate authority, and a digital signature for the certificate authority. A certificate 101c for device 101 and other nodes in a system 100 can comprise a certificate supporting X.509 v3 formats. The cryptographic parameters can specify parameters for both the device public key and also the digital signature from a certificate authority. Cryptographic parameters in a certificate can specify algorithms such as RSA or elliptic curve cryptography (ECC) values, ECC curve names, key lengths in bits, signature algorithms, validity dates, encoding rules, etc.

The signature of the certificate authority for cert.device 101c could be included in device certificate 101c. A node of system 100 in Figure 1, such as the authentication server function (AUSF) 103b can use the device certificate cert.device 101c in order to authenticate device 101 with MNO 102. Additional details regarding the use a certificate for device 101 of cert.device 101c will be described below in Figure 2. Note that other nodes in a system 100 can record and operate with a certificate similar to cert.device 101c, except with identities for the nodes, the public keys for the nodes, and certificate authority digital signatures for the public keys. A device 101 could record an use multiple different device certificates 101c (possibly with different device public keys corresponding to different device private keys), where the device 101 selects the different device certificates 101c based on the MNO 102 and/or the network 103 the device 101 communicates with.

A device 101 can record and use a device private key SK.device 101d (e.g. "secret key"). The private key can correspond to the public key in the device certificate cert.device 101c. In exemplary embodiments, device 101 internally generates a random number for the device private key SK.device 101d and using cryptographic algorithms 101s and generates the device public key in a certificate 101c. The private key SK.device 101d and the corresponding public key can be for the set of cryptographic parameters within the certificate 101c, such as the values for an elliptic curve named curve. The

generation of the public key for device certificate cert.device 101c could be (i) during device 101 manufacturing or distribution and (ii) before device 101 sends a SUCI 101b to a MNO 102.

In some exemplary embodiments before device 101 establishes a radio connection with MNO 102 (such as during a device configuration step), device provider 104 receives the public key for SK.device 101d and signs the public key with a device provider secret key such as SK.Provider 104e in order to generate the device certificate cert.device 101c. Or, a device certificate cert.device 101c could be signed by a different certificate issuer than a certificate issuer operated by device provider 104. Note that cryptographic algorithms 101s and PKI keys for device 101 can support post-quantum cryptography (PQC) and support code-based, lattice-based, and related cryptographic algorithms. Example possible algorithms for parameters in device certificate cert.device 101c include those described in “Post-Quantum Cryptography Standardization” from Wikipedia dated March 12, 2019, which is herein incorporated by reference.

Device 101 can store or record a public key for the device provider 104 of PK-ECIES.Provider 101e in order to conduct ECIES 101j encryption of the user name for a SUCI 101b. The public key can be associated with a set of cryptographic parameters ECIES Parameters 101f for using the public key (such as an ECC curve name), and the cryptographic parameters could be used with ECIES 101j function. In other words, ECIES Parameters 101f can specify a named curve associated with PK-ECIES.Provider 101e. The device provider 104 can record a private key corresponding to PK-ECIES.Provider 101e, where the private key can comprise SK-ECIES.Provider 104b as depicted in Figure 1. The value of PK-ECIES.Provider 101e could be written to memory 101m during manufacturing or distribution or configuration of device 101. Note that PK-ECIES.Provider 101e can comprise multiple different points on different ECC curves, such that different values of PK-ECIES.Provider 101e can be selected depending on the parameters ECIES Parameters 101f used by ECIES 101j. In other words, PK-ECIES.Provider 101e can include a first public key for name curve P-256, a second public key for named curve 25519, where the selection of the first public key or second public key could be based on ECIES Parameters 101f.

MNO parameters 101g can include a list of numbers or strings for values such as (i) allowed frequencies or frequency bands to scan, (ii) preferred access lists for roaming onto other wireless networks, (iii) criteria for a device 101 to select MNO 102, including in idle mode, (iv) support for emergency services, (v) supported languages or character encoding, (vi) codes to search for in beacons broadcast by a wireless network 102, (vii) parameters for a radio 101r to use when connecting to a wireless network 102, (viii) names or addresses for a server associated with a MNO 102 in order for a device 101 to conduct radio resource connect procedures, etc. A device 101 can record multiple sets of MNO parameters 101g for a plurality of different MNO 102, and a device 101 could select a MNO 102 from a plurality of available MNO 102 for radio 101r based on data within MNO parameters 101g.

Device 101 can derive an ephemeral public key ePK.ECIES.Device 101h and an associated secret key or private key eSK.ECIES.Device101i, using a random number, a key pair generation function, and the set of ECIES Parameters 101f. Device 101 can utilize a random number generator similar such as hardware random number generator in order to derive the private key eSK.ECIES.Device101i and input the random number into the key pair generation function. A key pair generation function could be included in application software running on device 101, and exemplary software for key pair generation algorithm can comprise OpenSSL, crypto++, or Mozilla libraries. The resulting ephemeral public key ePK.ECIES.Device 101h can be sent to MNO 102 in a message 205 below in Figure 2. The resulting secret key or private key eSK.ECIES.Device101i can be input into the encryption function ECIES 101j in order to convert a SUPI 101a into a SUCI 101b, where the user portion of an NAI for a SUPI 101a can be encrypted in a SUCI 101b. In some embodiments, the realm portion of the NAI in a SUCI 101b can remain plaintext, such that wireless network 102 can read the plaintext realm for the SUCI 101b in order to select and forward the SUCI 101b to a network 103 and/or device provider 104.

Device 101 can include an encryption function ECIES 101j in order to convert a SUPI 101a into a SUCI 101b. The encryption function ECIES 101j can comprise the identity encryption scheme as described in Annex C: “Protection schemes for concealing the subscription permanent identifier” in ETSI standard TS 33.501 version 15.1.0. In summary, device 101 can conduct an elliptic curve Diffie-Hellman key exchange using private key eSK.ECIES.Device101i and public key PK-ECIES.Provider 101e and parameters ECIES Parameters 101f in order to generate an encryption key and a message authentication code (MAC) key. The user portion of SUPI 101a, or the user name, can be input into the encryption function ECIES 101j along with the derived encryption key and MAC key in order to generate SUCI 101b which contains the user name as a ciphertext (depicted as “{user}” in Figure 1 and Figure 2 below). A MAC code generated by ECIES 101j using the MAC key can be used to verify message integrity for the SUCI 101b by device provider 104. The realm portion of SUPI 101a can remain as plaintext for SUCI 101b after encryption function ECIES 101j.

Device 101 can record or store a certificate for device provider 104 comprising Cert.Provider 104c, where the certificate for device provider 104 can be similar to the certificate for device 101 described above. An identity for device provider 104 recorded with Cert.Provider 104c can comprise provider ID 104a. Cert.Provider 104c can also record a public key for device provider 104 comprising PK.Provider 104c-a. Cryptographic parameters can be included in the certificate for device provider 104, such as an ECC named curve for PK.Provider 104c-a, a time for validity of the certificate, and other information normally recorded in an X.509 v3 certificate. Note that in exemplary embodiments, the certificate Cert.Provider 104c can be associated with and generated by device provider 104 and not MNO 102 and network 103. As depicted and described below in connection with Figure 2, exemplary embodiments contemplate that (i) device 101 can authenticate with network 103 and (ii) MNO 102 can

obtain an anchor key for device 101 without device 101 receiving or storing a certificate for MNO 102 and network 103 different than the stored or recorded device provider certificate Cert.Provider 104c .

In this manner, the use of a device 101 in a system 100 can be significantly more flexible than conventional technology, since a device provider 104 can control the device 101 before connecting to MNO 102 or selecting MNO 102 and device 101 does not need to receive a separate server certificate that chains to (or can be verified through) a previously recorded root certificate Cert.CI-root 105a. In other words, without the technology described herein, an MNO 102 and network 103 operating an AUSF 103b may not readily guarantee (or without user interaction for device 101) that device 101 can verify a server certificate received from AUSF 103b that can be verified with a recorded root certificate Cert.CI-root 105a.

The certificate Cert.Provider 104c can be signed by a certificate authority such as a device provider certificate issuer, and device 101 can record a chain of certificate authority certificates up to a root certificate Cert.CI-root 105a for a certificate issuer 105. For example, as depicted in Figure 1, device 101 Cert.Provider 104c could be signed by a certificate issuer for device provider 104 with a certificate comprising Cert.Provider-CI 104f. The public key for the device provider certificate issuer in Cert.Provider-CI 104f could be signed by the certificate issuer 105. The certificates for device provider 104 including Cert.Provider 104c and the supporting chain of certificates including Cert.Provider-CI 104f and root certificate Cert.CI-root 105a could be written by device provider 104 during distribution or configuration of device 101 or before device 101 is received by an end user of device 101. The certificates for device provider 104 including Cert.Provider 104c and the supporting chain of certificates including Cert.Provider-CI 104f and root certificate Cert.CI-root 105a could be stored by device 101 before device 101 sends data 106 to a MNO 102 in order to authenticate the device 101 with MNO 102 .

For embodiments where the certificate Cert.Provider 104c is written to device 101 or stored by device 101 before a device 101 connects with a MNO 102, then a device 101 could also (i) receive Cert.Provider 104c in a message from network 103 and (ii) check that the received Cert.Provider 104c matches the stored Cert.Provider 104c, and in this manner Cert.Provider 104c could be authenticated without requiring device 101 to conduct a full set of certificate signature verification steps through parent certificates of Cert.Provider 104c up to root certificate Cert.CI-root 105a.

Note that without recording Cert.Provider 104c in memory before connecting to a MNO 102, then device 101 could receive a server/network certificate from AUSF 103b that does not chain to (or be verified through) root certificate Cert.CI-root 105a, which can cause significant problems with conventional technology. The reason is device 101 may not have access to IP network 107 without first authenticating with AUSF 103b and may have no ability to request a parent certificate from AUSF 103b that chains to (or can be verified through) root certificate Cert.CI-root 105a stored on device 101. Without (i) previously recording Cert.Provider 104c and (ii) verifying a received Cert.Provider 104c

through a root certificate Cert.CI-root 105a (or some other certificate previously securely stored in device 101 before connecting with MNO 102), then device 101 may not feasibly be able to verify a received Cert.Provider 104c and thus authenticate MNO 102 and network 103.

Device 101 can derive an ephemeral public key ePK.TLS.Device 101n and an associated secret key or private key eSK.TLS.Device 101q, using a random number, a key pair generation function, and the set of cryptographic parameters for a TLS session. Device 101 can utilize a random number generator similar such as hardware random number generator in order to derive private key eSK.TLS.Device 101q and input the random number into the key pair generation function. A key pair generation function could be included in application software running on device 101, and exemplary software for key pair generation algorithm can comprise OpenSSL, crypto++, or Mozilla libraries. The resulting ephemeral public key ePK.TLS.Device 101n can be sent to MNO 102 and network 103 in a message 218 below in Figure 2 in a TLS 1.3 “Client Hello” message.

Device 101 could use a set of cryptographic algorithms 101s in order to process and use ePK.TLS.Device 101n and eSK.TLS.Device 101q with TLS messages such as a “client hello”, a “server hello”, and subsequent TLS messages. Private key eSK.TLS.Device 101q and a public key for network 102 of ePK.TLS.Network 103n can be input into an ECDH key exchange algorithm within cryptographic algorithms 101sa in order to mutually derive a shared secret key. The shared secret key can be used to generate or derive a symmetric ciphering key and a MAC key. Device 101 could use the PKI keys with cryptographic algorithms 101s and cryptographic parameters 101x in order to mutually derive a symmetric ciphering key and a MAC key with network 102, as specified in IETF RFC 8446 titled “The Transport Layer Security (TLS) Protocol Version 1.3”, which is herein incorporated by reference in its entirety. Note that device 101 and network 103 could also use the PKI keys to mutually derive a symmetric ciphering key and a MAC key using previous versions of TLS as well, such as TLS version 1.2.

Cryptographic algorithms 101s can include the steps and logic for processor 101p in device 101 to conduct in order for device 101 to securely communicate with MNO 102 and network 103. Cryptographic algorithms 101s can include at least symmetric ciphering algorithms, a random number generator, a key pair generation algorithm, digital signature algorithms, asymmetric ciphering algorithms, and key exchange algorithms. Cryptographic algorithms 101s can also include a key verification step for verifying that a received public key is valid for an ECC curve. Cryptographic algorithms 101s can use libraries available from example cryptographic suites such as OpenSSL, crypto++, BouncyCastle, or Mozilla, and other possibilities exist as well without departing from the scope of the present disclosure. Cryptographic algorithms 101s can use inputs of keys such as public keys, private keys, and/or symmetric keys along with cryptographic parameters 101x in order to for device 101 to process cryptographic data including ciphertext, key exchanges, and digital signatures.

Cryptographic parameters 101x can specify values or settings for (i) conducting an ECDH or ECDHE key exchange, (ii) mutually deriving a shared secret, (iii) mutually deriving a symmetric ciphering key from the shared secret, (iv) using a symmetric ciphering algorithm with the symmetric ciphering key, and (v) using a digital signature algorithm. As contemplated herein, cryptographic parameters 101x may also be referred to as parameters 101x. Each of device 101, MNO 102, and network 103 and device provider 104 can record at least one compatible subset of parameters within a set of cryptographic parameters 101x. Parameters 101x can specify values for an elliptic curve cryptography (ECC) curve name, key length, key formatting (e.g. compressed or uncompressed), encoding rules, etc.

As contemplated herein, the parameters 101x and cryptographic algorithms 101s used with ECC PKI keys and a key exchange in the present invention can be compatible and substantially conform with ECC algorithms and keys as specified in (i) the IETF Request for Comments (RFC) 6090 titled “Fundamental Elliptic Curve Cryptography Algorithms”, and (ii) IETF RFC 5915 titled “Elliptic Curve Private Key Structure”, and also subsequent and related versions of these standards. Other possibilities exist as well for cryptographic parameters 101x without departing from the scope of the present invention.

For use of ECC algorithms, parameters 101x can specify elliptic curve names such as, but not limited to NIST P-256, sect283k1, sect283r1, sect409k1, sect409r1, and other possibilities exist as well. Further, elliptic curves that do not depend on curves currently specified by the National Institute of Standards and Technology (NIST) could be utilized as well, such as, but not limited to, Curve22519, curve448, or FourQ. Parameters 101x can specify domain parameters for nodes in system 100 to calculate values or numbers in a compatible manner, such as common base point G for use with ECC PKI key pairs and a defining equation for an elliptic curve.

Network 103 can comprise a collection of servers and also operate as a cloud service. As depicted for system 100 in Figure 1, network 103 can communicate with device 101, MNO 102, and device provider 104 through IP network 107. Network 103 can include a network identity of network ID 103a, which could comprise a domain name, a name, or a string to uniquely identify network 103 in a system 100. In exemplary embodiments, at least a portion of network ID 103a is included in the realm of device identity comprising a NAI for SUPI 101a. Network 103 can include at least one server 111 as depicted in Figure 1. Server 111 can include hardware components similar to those of a device 101 depicted in Figure 1, except generally with larger capacities appropriate for a server 111. Server 111 can also operate as a host computing environment with physical hardware for a virtual machine to operate as a guest computing environment. In an exemplary embodiment, the AUSF 103b can comprise a virtual machine operating on server 111.

A server 111 can include random access memory (RAM) 111a, storage memory 111b, at least one system bus 111c, and at least one network interface 111d. As within a server 111 operating in a

network 103, server 111 can include at least one processor 103p in order to store and record data as well as communicate with other nodes over an IP network 107, such as with MNO 102 and device provider 104 through an IP network 107. Processor 103p can also be referred to as a central processing unit (CPU) 103p. Processor 103p can comprise a general purpose processor appropriate for the computational requirements for a server 101, and may operate with multiple different processor cores, including field programmable gate arrays (FPGA).

A processor 103p can comprise exemplary ARM® based processors or an Intel® based processor such as belonging to the XEON® family of processors, and other possibilities exist as well. Processor 103p can utilize the system bus 111c to fetch instructions from RAM memory 111a or storage memory 111b within a server 111 and operate on the instruction. A processor 103p can include components such as registers, accumulators, and logic elements to add, subtract, multiply, and divide numerical values and record the results in memory. In exemplary embodiments, at least one processor 103p within server 111 can be used to conduct the steps and message flows depicted in Figure 2 below.

RAM 111a may comprise a random access memory for Server 111. RAM 111a can be a volatile memory providing rapid read/write memory access to processor 103p. RAM 111a could be located on a separate integrated circuit in server 111. The system bus 111c may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures including a data bus. System bus 111c connects components within server 111, such as transferring electrical signals between the components illustrated for a server 111. Server 111 can include multiple different versions of bus 111c to connect different components, including a first system bus 111c between CPU 103p and RAM 111a (which could be a memory bus), and a second system bus 111c between CPU 103p and network interface 111d, which could be a SPI bus, a PCI bus, or similar data busses.

Server 111 may also operate a network interface 111d, where network interface 111d can be used to connect and communicate with other nodes such as depicted in Figure 1 and also Figure 2 below. Network interface 111d can comprise a physical interface connected to system bus 111c for server 111. In exemplary embodiments, network interface 111d can comprise an Ethernet or fiber optic wired connection. Network interface 111d can connect server 111 to plurality of computing devices and other servers through networks such as the globally routable public Internet.

Nonvolatile memory 111b or “storage” 111b (which can also be referred to herein as “storage memory 111b”) within server 111 can comprise a non-volatile memory for long-term storage of data, including times when server 111 may be powered off. Storage memory 111b may be a NAND flash memory or a NOR flash memory and record firmware for server 111, such as a bootloader program and an operating system. Storage memory 111b can record long-term and non-volatile storage of data or files for server 111. In exemplary embodiments, the network identity 103a is recorded in storage

memory 111b when server 111 is powered off, and the network identity 103a along with a network database 103d are moved by CPU 103p into RAM 111a when server 111 powers on.

Storage memory 111b can operate as a traditional hard disk drive or a solid state drive (SSD), or as a storage area network (SAN). Storage memory 111b can also comprise a plurality of spinning hard disk drives in a redundant array of independent disks (RAID) configuration. Storage memory 111b may also be referred to as “server storage” and can include exemplary file systems of FAT16, FAT 32, NTFS, ext3, ext4, UDF, or similar file systems. As contemplated herein, the terms “memory 111b”, “storage memory 111b”, and “nonvolatile memory 111b” can be considered equivalent.

Network 103 can include an authentication server function AUSF 103b as specified in ETSI TS 133 501 V15.1.0, as well as subsequent or related versions of ETSI standards. In general, an authentication server function can communicate with a security anchor function SEAF 102b from MNO 102. AUSF 103b can receive a SUCI 101b for device 101 from SEAF 102b, where MNO 102 receives SUCI 101b from device 101 via a wireless network and radio 102r. Additional details for the communication and operation of AUSF 103b for network 103 is provided below in Figure 2. In exemplary embodiments, AUSF 103b and device 101 can support EAP-TLS authentication, where EAP-TLS authentication for device 101 using a wireless network is described in Annex B of ETSI TS 133 501 V15.1.0. AUSF 103b can send and receive messages with device 101 through MNO 102 and also send and receive messages with device provider 104 through IP network 107. Network 103 can include at least one processor 103p, where processor 103p can operate in a similar manner as a processor 102p described above for MNO 102. Processor 103p could be included in a server 111 and also within AUSF 103b. A network 103 can operate with a plurality of processors 103p and/or servers 111 in order to perform the steps and communicate the messages depicted and described for a network 103 in Figure 2 below.

Network 103 can derive an ephemeral public key ePK.TLS.Network 103n and an associated secret key or private key eSK.TLS.Network 103q, using a random number, a key pair generation function, and the set of cryptographic parameters for a TLS session. Network 103 can utilize a random number generator similar such as hardware random number generator in order to derive the private key eSK.TLS.Network 103q and input the random number into the key pair generation function. A key pair generation function could be included in application software running in network 103. The PKI keys depicted and associated cryptographic algorithms and cryptographic parameters could be operated within AUSF 103b, where AUSF 103b operates on a server 111. The cryptographic algorithms and cryptographic parameters operated within AUSF 103b could be equivalent to cryptographic algorithms 101s and cryptographic parameters 101x used by device 101. Exemplary software for key pair generation algorithms and cryptographic algorithms equivalent to cryptographic algorithms 101s within network 103 can comprise OpenSSL, crypto++, or Mozilla libraries.

The resulting ephemeral public key ePK.TLS.Network 103n can be sent to MNO 102 and device 101 in a message 227 below in Figure 2 in a TLS 1.3 “Server Hello” message as key share 103n. Network 103 could use a set of cryptographic algorithms in order to process and use ePK.TLS.Network 103n and eSK.TLS.Network 103q with TLS messages such as a “client hello”, a “server hello”, and subsequent TLS messages. Private key eSK.TLS.Network 103q and a public key for device 101 of ePK.TLS.Device 101n can be input into an ECDH key exchange algorithm in order to mutually derive a symmetric ciphering key and a MAC key. AUSF 103b could use the PKI keys with cryptographic algorithms and cryptographic parameters in order to mutually derive a shared secret key and then a symmetric ciphering key and a MAC key with device 101, as specified in IETF RFC 8446 titled “The Transport Layer Security (TLS) Protocol Version 1.3”, which is herein incorporated by reference in its entirety. Note that device 101 and network 103 could also use the PKI keys to mutually derive a symmetric ciphering key and a MAC key using previous versions of TLS as well, such as TLS version 1.2.

As depicted in Figure 1, a network 103 can also operate a network database 103d. A network database 103d could comprise a collection of servers and storage memory for storing values associated with a plurality of devices 101 in order to support the operation of a system 100. A network database 103d could store values, strings, or numbers for device 101 such as a SUCI 101b received from MNO 102 and a device provider certificate 104c for device 101 received from device provider 104, and other data could be stored in a network database 103d as well. Exemplary data for an exemplary network database 103d is depicted and describe in connection with Figure 3 below. Data within a network database 103d could be recorded or stored as network 103 receives data for a device 101 from a MNO 102 and a device provider 104. Subsequent communications after receipt of the data could use a network database 103d in order to select stored values for communication with device 101.

Device provider 104 can comprise an entity or set of servers for authenticating or managing plurality of devices 101. Both device provider 104 and MNO 102 could operate a plurality of server 111. Although depicted in Figure 1 and Figure 2 as “device provider 104”, an entity such as a device owner, a device manufacturer, or a device distributor could record the data and perform the functions of a “device provider 104”. In general, a device provider 104 can comprise an entity or collection of servers which is the source for providing Cert.Provider 104c to device 101 before device 101 connects with MNO 102, where Cert.Provider 104c can be used by device 101 to authenticate MNO 102 and with AUSF 103b below in Figure 2. As contemplated herein, a “device provider” may also be referred to as a “provider”. Consequently, a device provider secret key SK.Provider 104e can be referred to as a “provider secret key 104e”, a device provider certificate Cert.Provider 104c can be referred to as a “provider certificate 104c”, etc.

Although depicted as “device provider” in Figure 1 and Figure 2, the device provider 104 could be a device owner, a device distributor, or a device manufacturer. In general, a device provider has a

sufficient level of ownership or control of device 101 before device 101 connects with MNO 102 such that device provider 104 could record or store the device identity SUPI 101a, and at least a certificate 101c for device 101 before device 101 connects with MNO 102. For some embodiments, device provider 104 could also store a device provider certificate 104c before device 101 connects with MNO 102. For some embodiments, device provider 104 could also be a device user, where the device user operates a plurality of different devices 101. As one example, device provider 104 could be a large enterprise responsible for hundreds or more devices 101 that are geographically dispersed, and the enterprise may prefer to support the authentication and networking connectivity for device 101 using the steps and messages depicted for a system 200 in Figure 2 below.

Device provider 104 can include a provider identity of provider ID 104a. Provider ID 104a can comprise a name or unique identifier, including a domain name or at least a portion of a realm in a NAI, in order to uniquely identify device provider 104 in a system 100, since a system 100 could include a plurality of different device providers 104. At least a portion of the provider ID 104a can be included in the SUPI 101a and SUCI 101b, such as within the realm portion of the SUPI 101a and SUCI 101b, as depicted in Figure 1. Although a single provider ID 104a is depicted in Figure 1, a device provider could use a plurality of different related provider ID 104a names or identities, such as “company-1”, “company-2”, etc. The use of different provider ID 104a names is also depicted and described for in a device database 104d in Figure 3 below.

Different devices 101 could include the different provider ID 104a names in the SUCI 101b, such as a first set of devices using an exemplary value of “company-1” in the realm portion of a SUCI 101b and a second set of devices using an exemplary value of “company-2” in the realm portion of SUCI 101b. In exemplary embodiments, the provider ID 104a in the realm portion of a SUCI 101b can be used to select from device database 104d the SK-ECIES.Provider 104b for an ECIES decryption function 104j, such that different devices 101 can use different corresponding PK-ECIES.Provider 101e for a SUPI/ECIES encryption function 101j by the different devices. The use of different provider ID 104a names and different secret keys SK-ECIES.Provider 104b for device provider 104 is also depicted for a device database 104d in Figure 3 below.

Device provider 104 can store a plurality of device identities comprising SUPI 101a for a plurality of devices 101. Although depicted as external to device database 104d in Figure 1, the values of SUPI 101a could be recorded within device database 104d along with the unique user name portion for each of the devices 101. Note that for a given device 101 in a system 100, a user name may not be globally unique, but the combination of user name and realm for a SUPI 101a can be globally unique in preferred embodiments. Device provider 104 can also store the concealed identity SUCI 101b received from a network 103 and originated by device 101 in order to conduct an ECIES 104j decryption step to convert the SUCI 101b into a SUPI 101a for a device 101.

Device provider 104 can also store or record a plurality of device certificates 101c corresponding to a plurality of devices in a system 100. The device provider 104 could be responsible for initial configuration of devices 101 or receive initial configuration data for devices 101, where the initial configuration of device 101 can include the generation of a device certificate 101c, and device provider 104 could store the device certificate 101c for each of the devices 101. Although depicted as external to a device database 101d, device provider 104 could store a plurality of device certificates 101c in a device database 101d. By device provider 104 storing a device certificate 101c and providing the device certificate 101c to a network 103, the authentication by network 103 of device 101 can be simplified, as described below in Figure 2 (since network 103 could trust device provider 104 and receive the device certificate 101c from device provider 104, where network 103 may not necessarily trust device 101 when a certificate 101c is received from device 101).

Device provider 104 can store or record a secret key for the device provider 104 of SK-ECIES.Provider 104b in order to conduct ECIES 104j decryption of the user name in a received SUCI 101b. The private key can be associated with a set of cryptographic parameters ECIES Parameters 101f for using the private key (such as an ECC curve name), and the cryptographic parameters could be used with ECIES 104j decryption function. Note that a device provider 104 could use (i) a first SK-ECIES.Provider 104b with a first provider ID 104a with ECIES 104j decryption function and (ii) a second SK-ECIES.Provider 104b with a second provider ID 104a with ECIES 104j decryption function, which is also depicted within a device database 104d in Figure 3 below.

ECIES Parameters 101f can specify a named curve associated with private key SK-ECIES.Provider 104b. As described above, the device 101 can record a public key corresponding to SK-ECIES.Provider 104b, where the public key for device 101 can comprise PK-ECIES.Provider 101e as depicted in Figure 1. The value of PK-ECIES.Provider 101e could be generated by device provider 104 using SK-ECIES.Provider 104b and ECIES Parameters 101f. PK-ECIES.Provider 101e could be provided to device 101 before device 101 sends a SUCI 101b to MNO 102, such as during device manufacturing or device distribution or device configuration. Different values of SK-ECIES.Provider 104b and corresponding ECIES Parameters 101f can be selected by device provider 104 using (i) at least a portion of provider ID 104a in a realm within a received SUCI 101b for device 101 and (ii) a device database 104d. SK-ECIES.Provider 104b can comprise a first private key for named curve P-256, a second public key for named curve 25519, where the selection of the first private key or second private key could be based on ECIES Parameters 101f, which could be associated with the provider ID 104a in a realm received in a SUCI 101b for device 101. Note that the transmission of a SUCI 101b by device 101 can also include a specified set of ECIES parameters 101f in order to identify a named curve or parameters associated with ePK-ECIES.Device 101h transmitted by device 101 along with the SUCI 101b.

Device provider 104 can include a decryption function ECIES 104j in order to convert a SUCI 101b into a SUPI 101a. The decryption function ECIES 104j can comprise the identity decryption scheme as described in Annex C: “Protection schemes for concealing the subscription permanent identifier” in ETSI standard TS 33.501 version 15.1.0. In summary, device provider can conduct an elliptic curve Diffie-Hellman key exchange using (i) private key SK-ECIES.Provider 104b, (ii) a received device public key ePK-ECIES.Device 101h received along with SUCI 101b, and (iii) parameters ECIES Parameters 101f in order to generate an encryption key and a message authentication code (MAC) key. The user portion of SUCI 101b, or the encrypted user name, can be input into the decryption function ECIES 104j along with the derived encryption key and MAC key in order to generate SUPI 101a which contains the user name as a plaintext. A MAC code generated by ECIES 104j using the MAC key can be used to verify message integrity for the SUCI 101b by device provider 104. The realm portion of SUCI 101b can be received as plaintext and may not be altered or changed by ECIES 104j. In exemplary embodiments, a MAC code output by ECIES 104j is calculated over the entire SUCI 101b, including an encrypted user name for device 101, and thus the integrity or authentication of a received plaintext realm within the SUCI 101b can be verified by comparing the received MAC code for a SUCI 101b and a calculated MAC code output the ECIES 104j.

A certificate for device provider 104 can comprise Cert.Provider 104c and can include the device provider identity comprising provider ID 104a. A certificate 104c for device provider 104 can include a public key corresponding to the device provider private key SK.Provider 104e, a set of cryptographic parameters for the device provider public key, the identity of a certificate authority, and a digital signature for the certificate authority, as well as a specified digital signature algorithm used for the digital signature. A certificate 104c for device provider 104 and other nodes in a system 100 can comprise a certificate supporting X.509 v3 formats, and other formats or structures for a device provider certificate Cert.Provider 104c and other certificates as contemplated herein are possible as well. The cryptographic parameters can specify parameters for both the device provider public key and also the digital signature from a certificate authority. Cryptographic parameters in a certificate can specify algorithms such as RSA or elliptic curve cryptography (ECC) values, ECC curve names, key lengths in bits, signature algorithms, validity dates, encoding rules, etc. The signature of the certificate authority or certificate issuer for Cert.Provider 104c could be included in device provider certificate 104c.

Note that a device provider 104 could use a plurality of different device provider certificates 104c, and the device provider certificate 104c for a device 101 could be selected from a device database 104d for a device 101 using the SUPI 101a output from decryption function ECIES 104j. In some exemplary embodiments, the device provider certificate Cert.Provider 104c is also recorded in a device 101 as depicted in Figure 1 before device 101 sends a SUCI 101b to MNO 102, such as during a device configuration step or a device manufacturing step. In other exemplary embodiments, device 101 may not record Cert.Provider 104c before device 101 sends a SUCI 101b to MNO 102, and for these

embodiments Cert.Provider 104c could be received and verified by device 101 in a TLS handshake, where device 101 could verify Cert.Provider 104c using another “parent” certificate for Cert.Provider 104c such as certificate issuer 105 root certificate Cert.CI-root 105a.

A device provider 104 can record and use a device provider private key SK.Provider 104e (e.g. “secret key”). The private key can correspond to the public key in the device provider certificate Cert.Provider 104c. In exemplary embodiments, device provider 104 (i) internally generates a random number for the device provider private key SK.Provider 104e and using cryptographic algorithms 104s, (ii) generates the device provider public key in a Cert.Provider 104c, and (iii) stores a copy of a set of cryptographic parameters within the certificate Cert.Provider 104c, such as the values for an elliptic curve named curve and specification of digital signature algorithms. The generation of the public key for device provider certificate Cert.Provider 104c could be before or during device 101 manufacturing or distribution and before device 101 sends a SUCI 101b to a MNO 102. In this manner, a device provider certificate Cert.Provider 104c could be written to device 101 before device 101 sends a SUCI 101b to MNO 102.

In some exemplary embodiments, device provider 104 operates a parent or “intermediate” certificate authority or certificate issuer in order to generate and sign Cert.Provider 104c. The public key for SK.Provider 104e in Cert.Provider 104c could be signed by a device provider secret key SK.Provider-CI 104g for Cert.Provider-CI 104f. The public key for Cert.Provider-CI 104f could be signed by SK.Root 105b from certificate issuer 105, where the certificate authority/issuer signature in Cert.Provider-CI 104f can be verified with root certificate Cert.CI-root 105a which can be stored in a device 101. Although a single level of a certificate issuer between device provider certificate Cert.Provider 104c and root certificate Cert.CI-root 105a is depicted in Figure 1, a device provider 104 could operate multiple levels of intermediate certificates between device provider certificate Cert.Provider 104c and root certificate Cert.CI-root 105a. In addition, although a single root certificate issuer 105 is depicted in Figure 1, a system 100 could include multiple root certificate issuers 105, and device provider 104 could record multiple certificates Cert.Provider-CI 104f or Cert.Provider 104c signed by different root certificate issuers 105 with different SK.Root 105b from the different certificate issuers 105.

Cryptographic algorithms 104s can include the steps and logic for processor 104p in device provider 104 to conduct in order for device provider 104 to securely communicate with network 103. Note that a processor 104p can be equivalent to a processor 103p for a server 111 above, and a device provider 104 can operate a plurality of servers 111. Cryptographic algorithms can include symmetric ciphering algorithms, a random number generator, a key pair generation algorithm, digital signature algorithms, asymmetric ciphering algorithms, and key exchange algorithms. Cryptographic algorithms can also include a key verification step for verifying that a received public key is valid for an ECC curve. Cryptographic algorithms 104s can use libraries available from example cryptographic suites

such as OpenSSL, crypto++, BouncyCastle, or Mozilla, and other possibilities exist as well without departing from the scope of the present invention. Cryptographic algorithms 104s can use inputs of keys such as public keys, private keys, and/or symmetric keys along with cryptographic parameters 104x in order to for device provider 104 to process cryptographic data including ciphertext, key exchanges, and digital signatures.

Cryptographic parameters 104x can specify values or settings for (i) conducting an ECDH or ECDHE key exchange to obtain or derive a shared secret, (ii) mutually deriving a symmetric ciphering key using at least the derived shared secret, (iii) using a symmetric ciphering algorithm with the symmetric ciphering key, and (iv) using a digital signature algorithm. Cryptographic algorithms 104s for device provider 104 can correspond or be equivalent to cryptographic algorithms 101s for device 101, and cryptographic parameters 104x for device provider 104 can correspond or be equivalent to cryptographic parameters 101x for device 101. Note that a server 111 for a network 103 or device provider 104 can store and operate with cryptographic parameters equivalent to the depicted cryptographic parameters 104x.

Device provider 104 can operate a plurality of servers with at least one processor 104p. A processor 104p can comprise a central processing unit (CPU) or a “system on a chip” and be similar to a processor 102p for a MNO 102 described above. Device provider user interface 104u can comprise a web-based interface for administrators of device provider 104 to manage the function and operation of device provider 104 as well as manage a plurality of devices 101. User interface 104u can support the addition of new devices 101 in a system 100, the deletion or revocation of existing devices 101 in a system 100, etc. User interface 104u can also allow changes in values and tables for a plurality of devices 101 in a device database 104d, where a device database 104d is depicted and described in connection with Figure 3 below.

Device database 104d can record a plurality of values for a plurality of devices 101 in order for device provider 104 to manage or operate communications with device 101 through a network 103. Device database 104d can record for each device exemplary values of SUPI 101a, device certificate 101c, the provider ID 104a used by device 101, a device provider certificate 104c used by a device 101, etc. Additional details regarding the data stored in a device database 104d are depicted and described in connection with Figure 3 below.

Although Figure 1 depicts MNO 102, network 103, and device provider 104 as separate nodes or entities, in some exemplary embodiments different entities or function of the nodes could be combined. In one embodiment, a MNO 102 and a network 103 could be combined, such that a MNO 102 controls or operates the network 103. In this embodiment, then a MNO 102 would operate both a SEAF 102b and an AUSF 103b. Or, a network 103 could control or operate a MNO 102, such that the network 103 controls or operates both a SEAF 102b and an AUSF 103b. In addition, although a MNO 102 is depicted in Figure 1, the wireless networking technology operated by MNO 102 with radios 102r

could use WiFi based technology (e.g. based on 802.11 standards) and a collection of geographically distributed WiFi access points. The WiFi access points could be controlled by network 103 and a security anchor function 102b could be omitted. In another embodiment, the function and operation of a network 103 and a device provider 104 could combined, such that either (i) network 103 records the data and operates the function of a device provider 104, or (ii) a device provider 104 operates an AUSF 103b. Other possibilities exist as well for the combination of the data and functions for a MNO 102, network 103, and device provider 104 without departing from the scope of the present invention.

Figure 2

Figure 2 is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device, a mobile network operator, a network, and a device provider, in accordance with exemplary embodiments. System 200 can include a device 101, a mobile network operator (MNO) 102, a network 103, and a device provider 104. The nodes or entities within system 200 were also depicted and described in connection with Figure 1 above, where Figure 2 depicts exemplary steps for the nodes and exemplary message flows between the nodes. Although a single device 101, MNO 102, network 103, and device provider 104 are depicted in a system 200, a system 200 could include a plurality of each of the depicted nodes connected via an IP network 107. In addition, data recorded for device 101 and device provider 104 depicted and described above in Figure 1 (other than ephemeral PKI keys) can be recorded in the device 101 and device provider 104 depicted in Figure 2 before the steps and message flows are conducted in Figure 2.

In order to support authentication of devices 101 with network 103, a MNO 102 could establish a secure session 201 with network 103. Secure session 201 could comprise establishing a secure communications link between the two servers using protocols such as TLS, IPSec, a virtual private network (VPN), a secure shell (SSH), or similar networking, transport, or application layer technologies in order to establish secure communications between MNO 102 and network 103. Secure session 201 can utilize certificates for each of MNO 102 and network 103 in order to provide mutual authentication and mutual key derivation for a symmetric encryption key in secure session 201. Secure session 201 can also be conducted over IP network 107, although the secure session 201 could also be established or conducted through a private network. Other possibilities exist as well for establishing a secure session 201 between MNO 102 and network 103 without departing from the scope of the present disclosure.

In order to support authentication of devices 101 with network 103, a network 103 and device provider 104 could establish a secure session 202. Secure session 202 can be equivalent to secure session 201 described in the paragraph above. Secure session 202 could use certificates from both network 103 and device provider 104 for mutual authentication and the secure derivation of encryption keys for the secure session. Note that subsequent data transmitted from device provider 104 to network

103 through the secure session 202 can be trusted by network 103 on the basis of the trust for secure session 202. For example, network 103 and AUSF 103b could receive a device certificate 101c for a device 101, where the device certificate 101c may not be authenticated with a certificate chain and a root certificate recorded or used by network 103. However, network 103 could still trust the device certificate 101c since the device certificate 101c could be received through secure session 202, where secure session 202 can be mutually authenticated using certificates for both network 103 and device provider 104. In some exemplary embodiments, both secure session 201 and secure session 202 are established using TLS 1.3.

At step 203, device 101 can power on and load firmware and configuration data in order to connect with wireless network 102. A step 203 can also include firmware or software in device 101 loading a RF module or radio 101r within device with an operating configuration such as MNO parameters 101g in Figure 1 (such as frequencies to use, wireless technology to use such as 5G or WiFi 6, etc.). Step 203 can include device 101 scanning for available mobile networks 102 and then selecting a preferred mobile network. In exemplary embodiments, device 101 scans for mobile network identities broadcast in system information blocks, where the mobile network identity such as mobile network ID 102a matches the value for a mobile network in a realm for a SUPI 101a recorded by device 101.

In other words, device 101 could include multiple SUPI 101a with different realms for different networks 103 and device providers 104, and search for a MNO 102 in an RF beacon or signal that matches a realm for one of the multiple SUPI 101a. For other embodiments, the MNO ID 102a may not be included in a portion of a realm within a SUPI 101a, and a device 101 could select the MNO 102 using MNO parameters 101g. At the conclusion of a step 203, device 101 can select and store a SUPI 101a to use with a MNO 102. A step 203 can also include device 101 selecting MNO parameters 101g based on an observed and available MNO 102 for a radio 101r in device 101. The selection of a SUPI 101a for device 101 in a step 203 can include the selection of (i) a device provider 104 provider ID 104a for a realm of the SUPI 101a and (ii) a selection of network identity network ID 103a for a realm of SUPI 101a. The values for PK-ECIES.Provider 101e and ECIES Parameters 101f can be selected in a step 203 based on the provider ID 104a for the SUPI 101a. As one example a SUPI 101a selected in a step 203 could comprise an exemplary value of “XXXXXX@company.aws.com”, where XXXXXX is the user name, “company” is the provider ID 104a and “aws.com” is the network ID 103a.

At step 204, device 101 can conduct a radio resource connection (RRC) request with MNO 102 using a radio 101r and the selected MNO parameters 101g. The RRC can transition a radio 101r in device 101 from an idle state to an active state. In the active state, device 101 can transmit messages or data to MNO 102. A step 204 can also include device 101 generating an ephemeral PKI key pair of ePK.ECIES.Device 101h and eSK.ECIES.Device 101i. At step 101j, device 101 can conduct the ECIES 101j encryption function with secret key eSK.ECIES.Device101i and public key PK-ECIES.Provider 101e in order to derive a symmetric ciphering key and an MAC key. The user name

portion of a SUPI 101a can be input into the ECIES 101j along with the symmetric key and the MAC key in order to output a first portion the SUCI 101b comprising an encrypted user name for the device 101.

The realm portion of a SUPI 101a can be input into the ECIES 101j along with the MAC key in order to output a MAC code covering the realm portion the SUCI 101b, such that the realm portion can be verified using the MAC code. Or, a single MAC code for both the first portion of the SUCI 101b and the realm portion of the SUCI 101b could be generated or calculated in a step 204. As depicted in Figure 2, the combination of an encrypted user name, selected provider ID 104a for a realm, and the selected network ID 103a for a realm can comprise a value 204a. The value 204a can also include a MAC code using the MAC key from ECIES 101j encryption function, such that a device provider 104 receiving and decrypting the value 204a comprising at least the SUCI 101b and the MAC code can verify the integrity of the value 204a.

Device 101 can then send the selected MNO 102 through (i) a wireless network operated by MNO 102 and (ii) the radio resource connection from step 203 a message 205, where the message 205 includes the public key ePK.ECIES.Device 101h and the value 204a. MNO 102 and SEAF 102b can receive the message 205 and conduct a step 206 in order to process the message 205. At step 206, SEAF 102b can (i) read the plaintext realm for SUCI 101b in value 204a, and (ii) select the network 103 using the network ID 103a in a realm for SUCI 101b. In a preferred exemplary embodiment as depicted in Figure 2, the suffix of the realm can comprise the network ID 103a, but in other embodiments the prefix of the realm could comprise the network ID 103a.

Note that MNO 102 and network 103 could have prior commercial and technical agreements for MNO 102 to forward device 101 authentication data to network 103 and AUSF 103b. If MNO 102 has no commercial relationship with network 103, or secure session 201 could not be established, or MNO 102 cannot forward the data in message 205, then MNO 102 can send device 101 an error message with an error code explaining the failure. Device 101 could then take corrective steps, such as selecting a different MNO 102 or different network 103 via a different SUPI 101a for device 101 in order to send a message 205 to the different MNO 102.

MNO 102 can then send network 103 and AUSF 103b a message 207 through secure session 201, where (i) message 207 includes ePK.ECIES.Device 101h and the value 204a, and (ii) the value 204a includes SUCI 101b. The AUSF 103b for network 103 can receive the message 207. At step 208, the AUSF 103b can read and process the plaintext data in the realm for SUCI 101b in value 204a. At step 208, the AUSF 103b can select the device provider 104 using the device provider ID 104a in the realm for SUCI 101b. In a preferred exemplary embodiment as depicted in Figure 2, the prefix of the realm can comprise the provider ID 104a.

Note that network 103 and device provider 104 could have prior commercial and technical agreements for network 103 to forward device 101 authentication data to device provider 104. If

network 103 has no commercial relationship with device provider 104, or secure session 202 could not be established, or network 103 cannot forward the data in message 207, then network 103 can send MNO 102 an error message with an error code explaining the failure. MNO 102 could then send device 101 an error code explaining the failure, such that device 101 could take corrective steps, such as selecting a different MNO 102 or different network 103 in order to send a message 205 to the different MNO 102. A step 208 can include AUSF 103b storing SUCI 101b in a network database 103d for subsequent use in identifying device 101, such as when receiving a “client hello” message 218 below with a SUCI 101b added to message 218 by SEAF 102b.

Network 103 and AUSF 103b can then send a message 209 to device provider 104 through secure session 202, where message 209 can include (i) MNO ID 102a, and (ii) ePK.ECIES.Device 101h and the value 204a, where the value 204a includes SUCI 101b. Note that the inclusion of MNO ID 102a can be useful for device provider 104 in a message 209, since (i) device 101 could connect with multiple different MNO 102, and (ii) network 103 could support multiple different MNO 102, and (iii) device provider 104 may need to know which MNO 102 is used by device 101 for billing or administrative purposes. In other words, without including MNO ID 102a in message 209 then device provider 104 may not know which MNO 102 that device 101 attempts to connect with.

Device provider 104 can then conduct a series of steps in order to process the message 209. At step 210, device provider 104 can read the plaintext provider ID 104a in SUCI 101b and use device database 104d in order to select (i) secret key SK-ECIES.Provider 104b and (ii) ECIES Parameters 101f. As depicted in Figure 3 below for a device database 104d, a provider ID 104a can be uniquely associated with secret key SK-ECIES.Provider 104b and (ii) ECIES Parameters 101f. Device provider 104 can then conduct a verification of the received ePK.ECIES.Device 101h using the ECIES Parameters 101f. The verification step can comprise verifying that the received public key is on the curve specified by ECIES Parameters 101f, the public key value does not comprise the “point at infinity”, etc.

Device provider 104 can then conduct a step 104j' using ECIES decryption function 104j in order to convert the ciphertext user name in SUCI 101b into plaintext. ECIES 104j was depicted and described in connection with Figure 1 above. A step 104j' can also include deriving a MAC key and verifying a MAC code within value 204a in order to check the message integrity of SUCI 101b. The check of message integrity of SUCI 101b in a step 104j' can include verifying the message integrity of a plaintext realm portion of the SUCI 101b using a MAC key from ECIES 104j. In other words, device provider can securely store a SUPI 101a for device 101 after step 104j'. Device provider 104 can conduct a step 211 with the plaintext user name to query a device database 104d for data pertaining to device 101, and a step 211 can include verifying that device 101 with the SUPI 101a is authorized to connect with MNO 102 and/or network 103. In a step 211, device provider 104 can determine the MNO 102 that device 101 attempts to connect with based on the MNO ID 102a received in a message 209 above. If device 101 is not authorized to connect with MNO 102, then device provider 104 could send

an error message to network 103 (for forwarding to device 101) and not proceed to the additional steps depicted in Figure 2.

Although an ECIES decryption function 104j is depicted in Figure 2 as being conducted by a device provider 104, in some exemplary embodiments a network 103 could conduct the ECIES decryption function 104j and send device provider 104 a message 209 with the SUPI 101a for a device 101 (instead of message 209 including a SUCI 101b). For these embodiments, then network 103 could store the values for secret key SK-ECIES.Provider 104b and parameters ECIES Parameters 101f, as well as operate the ECIES decryption function 104j. In other words, although Figure 1 depicts values for secret key SK-ECIES.Provider 104b and parameters ECIES Parameters 101f, as well as the ECIES decryption function 104j as within a device provider 104, a network 103 could store the data and perform a step 104j' in some exemplary embodiments. In some exemplary embodiments, network 103 could conduct the step 104j' (instead of device provider 104 as depicted in Figure 2) to decrypt the encrypted user portion within a SUCI 101b received from MNO 102 in a message 207 in order to generate a SUPI 101a for device 101. For the embodiment described in this paragraph, then a message 209 could include a SUPI 101a instead of a SUCI 101b, and network 103 could conduct the steps of 211 and 104j' instead of a device provider 104. For the embodiment described in this paragraph, then the depicted key of "PK-ECIES.Provider 101e" stored by device 101 can be associated with a network 103 instead of a device provider 104.

At step 212, device provider 104 can use the user name from SUPI 101a to query device database 104d for the device certificate cert.device 101c and also the device provider certificate cert.provider 104c recorded by device 101, as depicted and described above in connection with Figure 1. Note that a device certificate cert.device 101c can include either (i) the SUPI 101a as the device identity within the certificate or (ii) the user name portion from a NAI for device 101. For some exemplary embodiments, a step 212 could omit the selection of either device certificate 101c or device provider certificate 104c, and either certificate could be omitted in a subsequent message such as message 213.

A step 212 can also include device provider 104 collecting an OSCP stapling 212a through root certificate Cert.CI-root 105a. OSCP stapling is described in the TLS 1.3 RFC 8446, and OSCP stapling 212a provides an updated signature from parent certificate issuers that the certificate has not been revoked. In exemplary embodiments, OSCP stapling 212a is provided for cert.provider 104c, such that device 101 could verify the OSCP stapling 212a using the certificate chain for cert.provider 104c recorded by device 101. OSCP stapling 212a can be important for device 101 because device 101 may not have connectivity to IP network 107 without successfully conducting an EAP-TLS authentication with MNO 102, but connectivity to IP network 107 would be required to check certificate revocation lists without OSCP stapling 212a. In other words, without OSCP stapling 212a from a step 212, device 101 may not feasibly be able to check for certificate revocation for the device provider certificate

cert.provider 104c selected by device provider 104 in a step 212. Device 101 can receive cert.provider 104c in a subsequent “server hello” message in Figure 2.

Device provider 104 can then send network 103 and AUSF 103b a message 213 through secure session 202, where message 213 can include the selected device provider certificate Cert.Provider 104c from a step 212, device certificate Cert.Device 101c (possibly w/ SUPI 101a) from a step 212, SUCI 101b, and OSCP stapling 212a for device provider certificate 104c. Note that the inclusion of SUCI 101b with message 213 (and SUPI 101a in a Cert.Device 101c) can provide AUSF 103b the mapping between SUCI 101b and SUPI 101a for a device 101. Network 103 and AUSF 103b can receive the message 213 and conduct a step 214 to store (i) device provider certificate Cert.Provider 104c and (ii) device certificate Cert.Device 101c for the SUPI 101a and (iii) SUCI 101b used by device 101 in a network database 103d as depicted in Figure 3 below. In exemplary embodiments Cert.Provider 104c includes a subset of cryptographic parameters 104x and Cert.Device 101c includes a subset of cryptographic parameters 101x.

Network 103 could operate a network database 103d similar to device database 104d, where an exemplary network database 103d is depicted in Figure 3 below. Network 103 could store received certificates such as device provider certificate Cert.Provider 104c and device certificate Cert.Device 101c for a plurality of devices 101 along with the SUCI 101b for devices 101 received in messages 207 and SUPI 101a for devices 101 received in messages 213. AUSF 103b can determine the SUPI 101a for device 101 based on the SUPI 101a within a device certificate 101c, and both the values for SUPI 101a and string or file for device certificate 101c could be stored in a network database 103d as depicted in Figure 2 below. Or, in another exemplary embodiment the SUPI 101a could also be sent external to a device certificate in a message 213. Network 103 and AUSF 103b can then use a network database 103d to select device provider certificate Cert.Provider 104c and device certificate Cert.Device 101c in subsequent steps, such as a step 219a using the SUPI 101a or SUCI 101b for a device 101. In addition, a message 213 from device provider 104 to network 103 could include a user name or identity of device 101 separate from or instead of a SUPI 101a. In other words, the realm portion of a SUPI 101a is not required for a message 213 and a device identity in the form of a user name from a NAI for device 101 can be sufficient for network 103 to conduct subsequent steps and messages as depicted in Figure 2.

In addition, although message 213 with Cert.Provider 104c and Cert.Device 101c is depicted as being sent by device provider 104 after the receipt of a message 209, device provider 104 could send Cert.Provider 104c and Cert.Device 101c to network 103 before a message 209 and after the establishment of secure session 202. Device provider 104 could send Cert.Provider 104c and Cert.Device 101c for a plurality of different devices 101. By sending message 213 as depicted in Figure 2, then OSCP stapling 212a can be more current or more “fresh” for device 101. In addition, a network 103 could record a plurality of device provider certificates Cert.Provider 104c in a network database 103d before network 103 receives a message 213, and in this embodiment then a message 213 could

include an identifier of a device provider certificate Cert.Provider 104c (such as a certificate identity) instead of the full certificate as a *.pem file or equivalent. Further, a network 103 could record a plurality of device certificates Cert.Device 104c in a network database 103d before network 103 receives a message 213, and in this embodiment then a message 213 could include an identifier of a device certificate Cert.Device 101c (such as a certificate identity) instead of the full certificate as a *.pem file or equivalent.

Further, although device provider certificate Cert.Provider 104c and device certificate Cert.Device 101c are depicted in Figure 2 as being sent in the same message 213, a message 213 could be sent in parts, such as the data being transmitted via a physical interface in device provider 104 separately. A device provider certificate Cert.Provider 104c or device certificate Cert.Device 101c could be selected by device provider at a later point in time than depicted in Figure 2, such as device provider 104 sending device provider certificate Cert.Provider 104c and/or device certificate Cert.Device 101c after the receipt of a message 220 from network 103. A step 211 to select the device provider certificate Cert.Provider 104c or device certificate Cert.Device 101c could be conducted after device provider 104 receives a message 220. Other possibilities for the sequence and timing for a device provider 104 to send a network 103 a device provider certificate Cert.Provider 104c or device certificate Cert.Device 101c are possible as well without departing from the scope of the present disclosure.

Note that network 103 and AUSF 103b can trust Cert.Provider 104c and Cert.Device 101c, since the certificates can be delivered through the secure session 202. Or, in a step 214 network 103 could conduct a certificate verification of Cert.Provider 104c and Cert.Device 101c through a root certificate, such as Cert.CI-root 105a. Note that Cert.Provider 104c and Cert.Device 101c do not have to be verified through a root certificate stored by network 103, since the certificates are received through secure session 202, where secure session 202 can be established with other certificates for device provider 104 (and the other certificates for a secure session 202 could be verified through a different root certificate than the Cert.CI-root 105a recorded by device 101 as depicted in Figure 1). A step 214 can also include AUSF 103b verifying that AUSF 103b supports the cryptographic parameters 101x and 104x in the certificates received in a message 213. In other words, AUSF 103b can use a step 214 to verify that AUSF 103b can conduct a TLS 1.3 session using the parameters and certificates for Cert.Provider 104c and Cert.Device 101c received in a message 213.

At step 103q' AUSF 103b can generate an ephemeral private key for a TLS session with device 101 comprising eSK.TLS.Network 103q and the parameters within Cert.Provider 104c. In other words, AUSF 103b can read the named elliptic curve and key length from Cert.Provider 104c received in message 213 and generate a private key for the named curve and key length in a step 103q'. The use and function of an ephemeral secret key eSK.TLS.Network 103q was also described above in Figure 1. AUSF 103b can generate the corresponding public key ePK.TLS.Network 103n. A step 103q' could also be conducted after receiving a "client hello" message 218. AUSF 103b can then send device 101

an “EAP-TLS Start” message 215, which is also depicted as step 6 in Figure B.2.1.1-1 of 3GPP TS 33.501 version 15.1.0. Note that message 215 is sent to MNO 102 and MNO 102 can transmit message 215 through the radio resource connection established by device 101 in step 204 above. Where MNO 102 functions as a “pass through” entity for messages between network 103 and device 101, then the vertical line for MNO 102 in Figure 2 is omitted.

Device 101 can receive the “EAP-TLS” start message 215 and can conduct a series of steps in order to generate the “client hello” message. Although TLS version 1.3 according to IETF RFC 8446 is depicted in Figure 2, subsequent and related version of a transport layer security protocol could be used as well between device 101 and an AUSF 103b operating in network 103. In a step 216, device 101 using processor 101p can select parameters for a subsequent “client hello” message 218 using the cryptographic parameters in the stored device provider certificate 104c associated with a device provider ID 104a used in the realm of a SUCI 101b transmitted by device 101 in a message 205. In this manner, the set of cryptographic parameters for a “client hello” message 218 and a “server hello” message 227 can be reasonably assured to be compatible before the “client hello” message 218 is transmitted by device 101. In other words, in exemplary embodiments, a device 101 can store a device provider certificate 104c before device 101 sends a message 205, including storing the device provider certificate 104c during a configuration of device 101 or a secure session for device 101 before device 101 sends a message 205.

The selected parameters in a step 216 can include at least a digital signature algorithm such as the elliptic curve digital signature algorithm as supported by TLS version 1.3 (or another TLS version for messages 218 and 227) and also a named elliptic curve supported by the TLS version in subsequent messages 218 and 227. In exemplary embodiments, the set of cryptographic parameters selected from stored device provider certificate Cert.Provider 104c can be a subset of cryptographic parameters 101x for device 101. Or, in some exemplary embodiments device 101 could select the entire set or a larger subset of cryptographic parameters 101x in a step 216 in order to generate a “client hello” message 218. The selected set of cryptographic parameters for device 101 in a step 216 can include data for extensions 217a below. Device 101 could also use cryptographic algorithms 101s with the selected parameters from a step 216 in order to generate a “client hello” message 218. Device 101 could also store first list of cryptographic algorithms 101s and a second list of cryptographic parameters 101x for device 101 and generate extensions 217a using the first list and the second list.

In a step 101q', device 101 can generate device ephemeral private key eSK.TLS.Device 101q and the corresponding device ephemeral public key ePK.TLS.Device 101n using the selected set of cryptographic parameters from a step 216 above. The generation of ePK.TLS.Device 101n and eSK.TLS.Device 101q were also previously described above for device 101 in Figure 1. Note that the ephemeral private and public key for device 101 and the ephemeral private and public key for network 103 can support post-quantum cryptographic algorithms, as specified by cryptographic algorithms and

cryptographic parameters stored in the two nodes. At the conclusion of a step 101q', device 101 can conduct a step 217 in order to complete the processing or generation of the "client hello" message 218. The "client hello" message 218 generated in a step 217 can include a random number generated by device 101 using cryptographic algorithms 101s, a list of cipher suites supported by device 101 such as a symmetric ciphering algorithm and a cipher block chaining mode, the identity or data for a key derivation function including a hash function, and the list of supported extensions 217a according to section 4.2 of IETF RFC 8446. In exemplary embodiments, the data for a "client hello" message 218 can be generated in a step 217 using the set of cryptographic algorithms 101s in device 101 and parameters from device provider certificate Cert.Provider 104c.

Device 101 can process extensions 217a, where the extensions 217a can either (i) omit a certificate_authorities extension or (ii) include at least a certificate authority specified by cert.provider 104c as stored by device 101, or (iii) the identity for the root certificate issuer 105, where device 101 includes a root certificate cert.root 105a as depicted in Figure 1 above. For some exemplary embodiments, a device 101 can verify a subsequent certificate received from AUSF 103b as cert.provider 104c by simply comparing if the received cert.provider 104c from AUSF 103b matches the cert.provider 104c stored in memory 101m, as depicted in Figure 1 above. The previous sentence pertains to verifying a received cert.provider 104c from AUSF 103b is trusted, and a device 101 would use the public key in cert provider 104c in order to verify the digital signature in a "CertificateVerify" 222a below segment or data within a TLS handshake with MNO 102 and AUSF 103b. In this manner the verification through a certificate authority chain could be omitted in exemplary embodiments, thereby simplifying mutual authentication for device 101 and AUSF 103b/MNO 102. Note that device 101 may not have access to IP network 107 other than sending messages to SEAF 102b, and thus device 101 may not have the ability to verify a server certificate from AUSF 103b (different than device provider certificate 104c) that does not verify through an existing root certificate or certificate authority previously stored in device 101, if the received certificate from AUSF 103b does not have a chain terminating with a stored root certificate 105a.

Note that Cert.Provider 104c and device certificate Cert.Device101c can be stored by device 101 before device 101 sends "client hello" message 218, and also in a manner under the control of device provider 104 (including during any of device configuration, device manufacturing, or device distribution), and thus device provider 104 can take steps when processing device provider certificate Cert.Provider 104c and device certificate Cert.Device101c to ensure the specified cryptographic parameters within the certificates are compatible. Device provider certificate Cert.Provider 104c and device certificate Cert.Device101c can have compatible or equivalent sets of cryptographic parameters (such as specifying the same digital signature algorithm and mutually supported or equivalent named elliptic curves), and in this manner device 101 and AUSF 103b can be reasonably assured that (i) the

negotiation and use of cryptographic parameters in TLS messages such as messages 218 and 227 will be compatible and (ii) the negotiation of a TLS handshake in messages 218 and 227 will be successful.

Device provider can select or specify for device 101 cryptographic parameters including extensions in section 4.2 of TLS version 1.3 for cryptographic algorithms 101s (such as during a device configuration step to load device 101 with the data depicted and described in Figure 1). The selected or specified cryptographic parameters can be compatible with device provider certificate Cert.Provider 104c and device certificate Cert.Device101c. In this manner, a TLS session can be reasonably assured to successfully be negotiated between device 101 and AUSF 103b before the series of steps and message flows depicted in Figure 2, including a message 218, a message 227, and a message 233.

In other words, through the use of (i) device provider certificate Cert.Provider 104c for AUSF 103b as a server certificate and (ii) cryptographic algorithms 101s supporting device provider certificate Cert.Provider 104c for AUSF 103b, a TLS session can be assured to be successful. This contrasts with conventional technology, where if AUSF 103b selects and uses a server certificate not stored in device 101 then the negotiation of the TLS session is more likely to fail from both (i) incompatible sets of cryptographic parameters or cryptographic algorithms, and (ii) the server certificate may not be able to be verified through a certificate chain in device 101 up to a stored root certificate Cert.CI-root 105a, and (iii) device 101 may not have access to IP network 107 in order to obtain additional certificates that could verify a server certificate through a stored root certificate.

If a server certificate selected by AUSF 103b for use in a TLS session is both (i) not previously securely received and stored by device 101, and (ii) is authenticated or verified through a certificate chain with a different root certificate than Cert.CI-root 105a stored by device 101, then device 101 may not be able to verify the sever certificate selected by AUSF 103b (different than Cert.Provider 104c stored by device 101). The potential issues for AUSF 103b to select a server certificate different than device provider certificate Cert.Provider 104c stored by device 101 are compounded by the likely scenarios where (i) hundreds of different AUSF 103b may be used globally with different cryptographic parameters and cryptographic algorithms used for the server certificate and also (ii) server certificates with different verifying certificate chains up to different root certificates. The steps and message flows depicted in Figure 2, along with supporting data recorded as depicted in Figure 1, provide a solution to the significant number of issues that can arise when an AUSF 103b uses a server certificate different than device provider certificate Cert.Provider 104c. In summary to solve the many potential issues and problems of an AUSF 103b using a server certificate different than the device provider certificate Cert.Provider 104c stored by device 101, the device 101 can (i) store the device provider certificate Cert.Provider 104c before sending a SUCI 101b in a message 205, (ii) the AUSF 103b can receive the device provider certificate Cert.Provider 104c from a device provider 104 in a message 213, and (iii) use the device provider certificate Cert.Provider 104c when conducting a TLS session with device 101 as depicted in Figure 2.

Device 101 can then send the TLS “client hello” message 218 to MNO 102 and through SEAF 102b, where MNO 102 forwards the “client hello” message 218 to network 103 and AUSF 103b. The “client hello” message 218 through MNO 102 is also depicted as message 8 and 9 in Figure B.2.1.1-1 of 3GPP TS 33.501 version 15.1.0. A “client hello” message 218 can include a key share value of ePK.TLS.Device 101n, which is depicted in Figure 2. A “client hello” message 218 can include additional data for a “client hello” message as described in section 4.1.2 of IETF RFC 8446, including the data processed by device 101 in 216 and 217, such as selected extensions 217a. Note that MNO 102 and SEAF 102b can include a Nausf_UEAuthentication_Authenticate Request value with the message 218a, such that AUSF 103b can determine which device 101 the “client hello” message 218 is associated with. In exemplary embodiments and also as supported by 3GPP TS 33.501 version 15.1.0, the Nausf_UEAuthentication_Authenticate Request received with message 218a by AUSF 103b can include the SUCI 101b. As depicted in Figure 2, the SEAF 102b can include the SUCI 101b with the “client hello” in message 218a sent to AUSF 103b that includes the “client hello” message 218 from device 101.

AUSF 103b can receive message 218a and conduct a series of steps to process the “client hello”. AUSF 103b can conduct a verification 219a of the received ePK.TLS.Device 101n using the named curve specified by cryptographic parameters within the “client hello” message 218 within extensions 217a. The verification step can comprise verifying that the received public key is on the curve specified by the received cryptographic parameters in extensions 217a, the public key value does not comprise the “point at infinity”, etc. The verification step 219a can comprise the verification step within section 4.2.8.2. of IETF RFC 8446.

AUSF 103b can then conduct a step 219b in order to select at least device provider certificate Cert.Provider 104c from a network database 103d using the SUCI 101b received along with the “client hello” from SEAF 102b in a message 218a. As noted above in a step 214 for AUSF 103b, the AUSF 103b can previously store device provider certificate Cert.Provider 104c, device certificate Cert.Device 101c, SUPI 101a, and SUCI 101b for device 101 in network database 103d after receiving the message 213 above from device provider 104. A step 219b can also comprise AUSF 103b also selecting the device certificate Cert.Device 101c from a network database 103d using the SUCI 101b received along with the “client hello” from SEAF 102b, although the device certificate Cert.Device 101c could be selected at a later time, such as after the receipt of a message 233. An AUSF 103b can use a step 219b to select a device provider certificate 104c from a network database 103d using the SUCI 101b from message 218a for conducting an EAP-TLS authentication. The selected device provider certificate 104c from a step 219b can comprise the server certificate for an AUSF 103b to use when conducting an EAP-TLS authentication with device 101, such as the selected device provider certificate 104c comprising the server certificate below in a “server hello” within message 227.

At step 219c, AUSF 103b can further process the “client hello” by taking the steps specified in IETF RFC 8446, such as generating a server random number, selecting extensions 219d according to section 4.2 of RFC 8446 based on a subset of extensions 217a received in the “client hello” message 218, generating a certificate request 219e according to section 4.3.2 of RFC 8446, etc. The certificate request 219e from a step 219c can comprise a request for device 101 to mutually authenticate with AUSF 103b using at least a device certificate 101c. In exemplary embodiments, for security and commercial reasons, a device provider 104 may prefer to record the secret key for the public key in device provider certificate Cert.Provider 104c. For these embodiments, then AUSF 103b may not be able to internally generate the digital signature for a certificate verify according to section 4.4.3 of RFC 8446. AUSF 103b can then send device provider 104 a message 220, where message 220 include “key share” 103n, which comprises ePK.TLS.Network 103n, Extensions 219d, CertificateRequest 219e, and device provider certificate cert.provider 104c. Note that message 220 can also include identifying information for data within the message 220, such as a SUPI 101a (depicted) or a SUCI 101b, such that device provider 104 could track which device 101 the data within message 220 is associated with.

In exemplary embodiments, a CertificateRequest 219e processed by AUSF 103b can either (i) omit a certificate_authorities extension or (ii) use at least a certificate authority specified by cert.device 101c recorded for device 101 in a network database 103d and also recorded by device 101. An AUSF 103b can verify a subsequent certificate received by device 101 as cert.device 101c by comparing if the received cert.device 101c from device 101c matches the cert.device 101c received from device provider 104 in a message 213, and in this manner the verification through a certificate authority chain could be omitted in exemplary embodiments thereby simplifying mutual authentication for device 101 and AUSF 103b/MNO 102 while maintaining security. The sentence above pertains to verifying a received cert.device 101c from device 101 is trusted, and an AUSF 103b would use the public key in cert.device 101c in order to verify the digital signature in a “CertificateVerify” segment within a TLS handshake with device 101. “CertificateVerify” in Figure 2 and as described herein can comprise a “CertificateVerify” according to section 4.4.3 of RFC 8446.

Note that a message 220 from AUSF 103b to device provider 104 could omit device provider certificate cert.provider 104c since device provider 104 previous sent 104c to AUSF 103b, however including all the depicted elements of a message 220 in one package or string of data segments can help ensure the input for a digital signature by device provider 104 will match exactly the bit sequence and message/segment structure as the “message to sign” input into a digital signature algorithm operated by device provider 104 with SK.Provider 104e in order to generate below a “CertificateVerify” 222a digital signature. In other words, in exemplary embodiments, the contents of a message 220 can include all of the data for AUSF 103b to generate and send a “CertificateVerify” 222a digital signature in a subsequent “server hello” message 227.

Device provider 104 can receive the message 220 with identifying information for device 101 such as the SUPI 101a (depicted) or a SUCI 101b, and in a step 221 verify the selected extensions 219d are supported by device 101 using data recorded by a device 101 in a device database 101d. Device provider 104 could compare that selected extensions 219d are supported by cryptographic algorithms 101s and cryptographic parameters 101x stored in device 101. Note that although not depicted for a device database 101d in Figure 3 below, a device database 104d could store a first list of cryptographic algorithms 101s and a second list of cryptographic parameters 101x in a device database 104d and in a step 221 determine that the selected extensions 219d are supported by device 101 by querying for the first list and the second list in the device database 104d using the SUPI 101a or other identifying information for device 101 in message 220. In a related exemplary embodiment, device provider 104 could send network 103 and AUSF 103b the first list of cryptographic algorithms 101s and a second list of cryptographic parameters 101x for device 101 using SUCI 101b in a message 213 above. Network 103 could use the first list and the second list in order to select extensions 219d in a step 219b, where the selected extensions 219d can be (i) the set of extensions 217d from a “client hello” message 218 or a (ii) a subset of the extensions 217d.

Step 221 can also determine that the CertificateRequest 219b can be properly processed by device 101, such as including a certificate_authorized extension with at least one certificate authority certificate stored in device 101, such as the certificate authority specified for cert.device 101c. At step 222, device provider can generate a digital signature comprising a “CertificateVerify” for the data received in a message 220 using the secret key SK.Provider 104e corresponding to the public key in Cert.Provider 104c and also the cryptographic parameters for the digital signature as specified in Cert.Provider 104c or extensions 219d. The digital signature processed in a step 222 can comprise a “CertificateVerify” 222a. In a step 222, device provider 101 could select the secret key SK.Provider 104e from a device database 104d using the device provider certificate cert.provider 104c received in a message 220. As contemplated herein, the certificate 104c received in a message 220 could also comprise a unique certificate identity for the certificate 104c, such that device provider 104 can identify the device provider certificate cert.provider 104c using the unique certificate identity.

Device provider 104 can then send AUSF 103b a message 223 with a digital signature of “CertificateVerify” 222a, where the “CertificateVerify” 222a can be over the data sent by AUSF 103b in a message 220 (e.g., “key share” 103n, which comprises ePK.TLS.Network 103n, Extensions 219d, CertificateRequest 219c, and device provider certificate cert.provider 104c). In this manner from the combinations of a step 219c, message 220, and step 222, a AUSF 103b can (i) generate a “message to sign” for a “server hello” comprising key_share 103n, Extensions 219d, CertificateRequest 219b, Certificate 104c, and (ii) receive a digital signature of “CertificateVerify” 222a from device provider 104 over the “message to sign”. AUSF 103b can use the received digital signature of a “CertificateVerify” 222a in order to generate a complete “server hello” message 227 for a SUCI 101b

that could be successfully processed by device 101 using the authentication and encryption steps specified for TLS 1.3 in IETF RFC 8446.

Although Figure 2 depicts device provider 104 generating the digital signature for “CertificateVerify” 222a, in some exemplary embodiments (i) an AUSF 103b could generate the “CertificateVerify” 222a, and (ii) the messages 220, step 222, and message 223 could be omitted from a system 200. For this embodiment, then device provider 104 could send the private key SK.Provider 104e for device provider certificate Cert.Provider 104c through secure session 202, such as in a message 213 above. For this embodiment, a network 103 could store the private key SK.Provider 104e for device provider certificate Cert.Provider 104c in a network database 103d and select the private key SK.Provider 104e using the selected device provider certificate Cert.Provider 104c for a device 101 SUCI 101b, where the selection of a device provider certificate Cert.Provider 104c for a device 101 SUCI 101b was described above for a step 219b.

AUSF 103b can receive message 223 and conduct steps to generate a “server hello” response for the received “client hello” from device 101. In step 224, AUSF 103b could verify the digital signature for “CertificateVerify” 222a using the public key and parameters in device provider certificate 104c in order to confirm the digital signature is correct and can be properly processed by device 101 with the public key and parameters in device provider certificate 104c.

For some exemplary embodiments of a system 200, a network 103 could use a step 219b to select a server public key for use in an EAP-TLS authentication with device 101 in a subsequent “server hello” message 227 instead of using a received device provider certificate 104c. Network 103 could record and use a corresponding server private key for the server public key. For these embodiments, a network 103 or AUSF 103b could send a server certificate for the selected server public key in a message 220, instead of the data depicted for a message 220 in Figure 2. For these embodiments, a message 220 could comprise a “certificate signing request” for the server public key in the selected server certificate. The “certificate signing request” from network 103 for the selected server certificate/server public key could comprise a request as described in the Wikipedia article titled “Certificate Signing Request” from May 1, 2019, which is herein incorporated by reference in its entirety. Device provider 104 could receive the “certificate signing request” for the server public key in the server certificate in the message 227. Device provider could generate a new certificate for the server public key by signing with the device provider private secret key SK.Provider 104e. The device provider secret key SK.Provider 104e could be selected from a device database 104d using the realm portion of a SUPI 101a in a message 220. For these embodiments, device provider 104 could send network 103 and AUSF 103b the new certificate in a message 223. Network 103 could then include the new server certificate in a subsequent “server hello” message 227, where the “CertificateVerify” 222a could be a digital signature generated by network 103 using the server private key for the server public key in the new certificate, where the new certificate could be received from device provider 104 in response to the “Certificate Signing

Request”. For these embodiments, a device 101 could (i) verify a “server hello” message using the server public key in the new server certificate, and (ii) verify the new server certificate using the device provider certificate 104c stored in device 101, as depicted in Figure 1.

At step 225, AUSF 103b can conduct an ECDH key exchange with device ephemeral public key ePK.TLS.Device 101n received in message 218 and server ephemeral private key eSK.TLS.Network 103q derived in a step 103q’ in order to mutually derive shared secret 225a and then derive a symmetric ciphering key 225b and an associated MAC key. A step 225 can follow the steps and procedures as specified by section 7 of RFC 8446 for the shared secret 225a and then deriving the symmetric ciphering key 225b, although other possibilities exist as well for the derivation of a shared secret key 225a and a symmetric ciphering key 225b without departing from the scope of the present invention. As supported by section 7 of RFC 8446, (i) a first portion of a symmetric ciphering key 225b and an associated MAC key for encryption by AUSF 103b and decryption by device 101 can be different than (ii) a second portion of a symmetric ciphering key 225a and an associated MAC key for decryption by AUSF 103b and encryption by device 101. However, both device 101 and AUSF 103b can conduct a step 225 in order to derive both the first and second portions of a symmetric ciphering key 225b.

At step 225, AUSF 103b can derive a shared secret 225a with device 101 as a point with x and y coordinates, where the device ephemeral public key is raised to the power of the scalar ephemeral private key with a modulus according to the named curve to for the PKI keys in cryptographic parameters associated with the keys and also in extensions 219d selected in a step 219b. The x coordinate of the resulting point can comprise a shared secret 225a. AUSF 103b or network 103 could store the derived shared secret 225a and the symmetric ciphering key 225b from a step 225 in a network database 103d as depicted in Figure 3 below, along with the SUCI 101b received in a message 218 and/or the combination of SUCI 101b and SUPI 101a received in a message 213. The symmetric key 225b and the associated MAC key can be derived from the shared secret 225a using a key derivation function (KDF) or a secure hash algorithm mutually shared with device 101. The KDF and secure hash algorithm could be specified and shared between device 101 and AUSF 103b in extensions 217b and extensions 219d, such as a secure hash algorithm of SHA-256, SHA-384, etc.

Device 101 can take equivalent steps with the corresponding keys as used by AUSF 103b in order to derive the shared secret 225a and the symmetric key 225b and an associated MAC key, which is also described in section 7 of IETF RFC 8446. At step 226, AUSF 103b can complete the processing of a “server hello” message according to RFC 8446 by encrypting the values Extensions 219d, CertificateRequest 219e, Certificate 104c, OSCP stapling 212a (from message 213), and Certificate Verify 222a with the symmetric ciphering key 225b derived in a step 225.

Other data could be included in a “server hello” message as well without departing from the scope of the present invention. Although Annex B of 3GPP TS 33.501 version 15.1.0 describes the use of TLS version 1.2, a MNO 102 and SEAF 102b and AUSF 103b in a system 200 could support the use

of TLS version 1.3. Or, the sequence of steps and messages in Figure 2 for a system 200 could be adjusted to support TLS version 1.2 without departing from the scope of the present disclosure. Further, future versions of EAP-TLS authentication for 5G networks and 3GPP networks could support TLS version 1.3, where the future versions could be compatible with the steps and message flows depicted for a system 200 in Figure 2.

AUSF 103b can then send through MNO 102 the “server hello” message 227. Note that AUSF 103b can insert or add the SUCI 101b for device 101 external to the “server hello” message 227, such that SEAF 102b and MNO 102 can send the “server hello” message 227 to the correct device 101. In other words, MNO 102 can use the SUCI 101b as external to the “server hello” message 227 in order to route the message 227 to device 101 through base stations and at least one gnodeb with a radio 102r. The data within a “server hello” message is depicted in Figure 2 and also described above with a step 226. The bracket values of “{“ and “}” in for message 227 and other messages or data depicted in Figures herein depict that the ciphertext data is encrypted with the derived symmetric ciphering key, such as key 225b from a step 225 and also verified with a MAC code using a MAC key from a step 225. The depiction of the key 225b next to the brackets for message 227 indicate that the message is encrypted with at least a mutually derived symmetric ciphering key 225b and does not suggest or indicate that the key 225b is transmitted with the message.

Device 101 can receive the “server hello” message 227 and take steps in order to process the message. At step 219a, device 101 can verify the received ephemeral public key using the same steps as a step 219a for AUSF 103b above, which can also comprise the verification step 4.2.8.2 in RFC 8446. At step 228, device 101 can conduct an ECDH key exchange with device ephemeral private key eSK.TLS.Device 101q and server ephemeral public key ePK.TLS.Network 103n derived in a step 103q in order to mutually derive a shared secret and then a symmetric ciphering key and a MAC key. A step 228 can follow the steps and procedures as specified by section 7.4 of RFC 8446. At step 228, device 101 can derive a shared secret with AUSF 103b as a point with x and y coordinates, where the network/server ephemeral public key 103n is raised to the power of the scalar device ephemeral private key with a modulus according to the named curve to for the PKI keys. The x coordinate of the resulting point can comprise a shared secret 225a, which was mutually derived by the AUSF 103b in a step 225. The symmetric key 225b and the MAC key can be derived from the shared secret 225a using a secure hash function, and additional data shared between device 101 and AUSF 103b in messages 218 and 217 could be input into the secure hash function with the shared secret 225a. A step 228 can also comprise device 101 decrypting the ciphertext data from a message 227 using the derived symmetric ciphering key 225b and verifying message 227 integrity with the MAC code and the MAC key derived in step 228.

A step 229 can comprise device 101 verifying the digital signature in the digital signature for “CertificateVerify” 222a using the public key and parameters in device provider certificate 104c in order

to confirm the digital signature is correct. Device 101 could confirm the received value for “CertificateVerify” 222a matches a calculated value by device 101 for “CertificateVerify” 222a. The message or data input into the digital signature algorithm can comprise the data key_share 103n, Extensions 219d, CertificateRequest 219e, Certificate 104c, and OSCP stapling 212a, and also follow the steps for signature verification in RFC 8446. Other possibilities exist as well for the steps to verify a digital signature of a “CertificateVerify” 222a using the public key in a device provider certificate 104c without departing from the scope of the present disclosure.

A step 230 can comprise device 101 verifying that the received device provider certificate 104c matches the recorded device provider certificate 104c in memory 101m as depicted in Figure 1. Or, if device provider certificate 104c does not match the recorded device provider certificate 104c in memory 101m, then device 101 can conduct a series of steps to verify the received device provider certificate 104c up through at least one recorded and securely received certificate within device 101, such as a root certificate Cert.CI-root 105a. Note that in some embodiments, the sequence of steps 229 and 230 could be switched, and the order of other steps depicted as contemplated herein could be adjusted without departing from the scope of the present invention. In addition, a step 230 can comprise device 101 verifying an OSCP stapling 212a for embodiments where device provider certificate 104c does not match the recorded device provider certificate 104c. Further, a step 230 can comprise device 101 verifying an OSCP stapling 212a if the device provider certificate 104c matches the certificate 104c stored by device 101 in order to confirm the current validity status of the device provider certificate 104c (e.g. device 101 could determine that certificate 104c has not been revoked).

Note that the system 200 depicted in Figure 2 can avoid the requirement that device 101 needs to verify OSCP stapling 212a, for embodiments where device provider 104 generates “CertificateVerify” 222a in a step 222. The presence of a “CertificateVerify” 222a, where device provider 104 generates the “CertificateVerify” 222a in a step 222 can confirm the certificate cert.provider 104c is valid for device 101. If the cert.provider 104c was revoked, then device provider 104 and/or AUSF 103b could decline to provide “CertificateVerify” 222a to device 101. If another party than device provider 104 obtained the secret key SK.Provider 104e, the party would not feasibly be able to use the key since MNO 102 only sends SUCI 101b for device 101 to AUSF 103b, and thus device 101 would not authenticate with a different AUSF 103b as specified in the realm of the SUCI 101b.

At step 231, device 101 can select a device certificate 101c that is compatible with the parameters received within message 228, such as with extensions 219d. A device 101 could record a plurality of different device certificates 101c, such as supporting different cryptographic parameters for digital signature algorithms and also different certificate authorities. Device 101 could use the data for extensions 219d, such as a certificate authority specified in extensions 219d, in order to select device certificate 101c. In some exemplary embodiments, the certificate authority for device certificate 101c

can be device provider 104, where the digital signature for device certificate 101c is signed with the device provider certificate issuer secret key SK.Provider-CI 104g. In other words, in exemplary embodiments device 101 selects a device certificate 101c in a step 231 where the signature authority for device certificate 101c is the device provider certificate issuer for device provider 104. Or, in other embodiments, device 101 could record a single certificate 101c and AUSF 101c can verify the certificate 101c by comparing the certificate 101c received from device 101 with the certificate 101c received from device provider 104.

At step 232, device 101 can generate a digital signature for a "CertificateVerify" 232a over at least the selected device certificate 101c with the private key SK.Device 101d for the selected device certificate 101c using the parameters from extensions 219d in message 227. The private key SK.Device 101d is depicted and described above in connection with Figure 1. Device 101 could follow the steps for generating a "CertificateVerify" in section 4.4.3 of RFC 8446, and other possibilities exist as well without departing from the scope of the present disclosure. A step 232 can include device 101 encrypting the device certificate 101c and the "CertificateVerify" 232a into a ciphertext with the symmetric ciphering key 225b derived in a step 228 above, as well as generating a MAC code using the derived MAC key.

Device 101 can then MNO 102 a message 233, where message 233 includes the ciphertext from a step 232 with the encrypted device certificate 101c and the encrypted digital signature of "CertificateVerify" 232a. Note that MNO 102 can use the radio resource connection of device 101 in order to track or identify device 101, and MNO 102 using SEAF 102b can append the SUCI 101b originally sent by device 101 in a message 205 to the message 233, equivalent to MNO 102 or SEAF 102b adding SUCI 101b to message 218 above. In other words, MNO 102 can track the radio resource connection used by device 101 in order to (i) append or add SUCI 101b to message 218 (e.g. message 218a received by AUSF 103b) (ii) append or add SUCI 101b to message 233 received by AUSF 103b. In other words, AUSF 103b can receive the message 233 with the SUCI 101b added, although device 101 may not add SUCI 101b to message 233. Or, in some exemplary embodiments device 101 can add SUCI 101b to both message 218 and message 233. As described below, AUSF 103b can use a network database 103d to record data for sessions with device 101, such as the SUCI 101b and the device provider certificate 104c as well as the server ephemeral private key eSK.TLS.Network 103q and symmetric ciphering key 225b associated with a device 101 and TLS messages. As depicted for a network database 103d in Figure 3 below, a network 103 could also store the derived symmetric ciphering key 225b from a step 225, and MAC key as well (although a MAC key in database 103d is not depicted in Figure 3), in order to select symmetric ciphering key 225b using the SUCI 101b from MNO 102 and decrypt the ciphertext in a message 233.

In exemplary embodiments, the device provider certificate 104c received from device provider 104 by AUSF 103c can function as the server certificate for AUSF 103b with device 101. AUSF 103b

can use a plurality of different server certificates with a plurality of different devices, and the proper certificate for AUSF 103b to use could be tracked in a network database 103d and selected for a device 101 from messages via MNO 102 using the SUCI 101b added to the messages by MNO 102.

In a step 234, AUSF 103b can use the selected (x) symmetric key 225b from a network database 103d for (y) SUCI 101b received in a message 233 from MNO 102 in order to decrypt the ciphertext included in message 233. The plaintext values from a message 233 can comprise the device certificate 101c and the "CertificateVerify" 232a. "CertificateVerify" 232a can comprise a digital signature from device 101 over the plaintext data within message 233 (but potentially excluding the digital signature for "CertificateVerify" 232a. At step 234, AUSF 103b can compare that received device certificate 101c from device 101 is the same as the received device certificate from device provider 104 in a message 213 above. AUSF 103b can trust or verify the received device certificate 101c from device 101 if the received device certificate 101c from device 101 in a message 233 is the same as the device certificate 101c received from device provider 104, such as in a message 213 via secure session 202. Or, if the received device certificate 101c from device 101 is not the same as 101c from device provider 104 (or AUSF 103b does not receive a device certificate 101c from device provider 104), then AUSF 103b can verify cert.device 101c through a certificate chain up to a securely recorded certificate in AUSF 103b such as a root certificate Cert.CI-root 105a.

At step 235, AUSF 103b can verify the digital signature value for "CertificateVerify" 232a using the device public key in cert.device 101c as well as the signature parameters in cert.device 101c. AUSF 103b can confirm the received value for "CertificateVerify" 232a matches an internally calculated value for "CertificateVerify" 232a. After successful device authentication in a step 235, AUSF 103b can conduct a step 236 in order to generate an anchor key for SEAF 102b and MNO 102. With EAP-TLS authentication between device 101 and AUSF 103b, the AUSF 103b and device 101 can mutually derive a key Kausf using the steps described in Annex B, section B.3 titled "Key Derivation" for 3GPP TS 33.501 version 15.1.0. AUSF 103b can conduct the steps to derive Kausf in a step 236. Although section B.3 describes the derivation of a Kausf in the context of EAP-TLS authentication using TLS version 1.2 with IETF RFC 5216, the equivalent steps can be conducted with a future version of EAP-TLS authentication supporting TLS version 1.3.

In summary, the successful handshake of messages 218, 227, and/or 233 can be used in a step 236 to derive a TLS master secret as described in section 7.1 of RFC 8446. Data input into the derivation of a TLS master secret for AUSF 103b can comprise the shared secret 225a derived in a step 225 and the plaintext values for data transmitted in message 218 and 227. The shared secret 225a derived in a step 225 and the plaintext values for data transmitted in message 218 and 227 can be input in a step 236 into a key derivation function using a secure hash algorithm in order to derive a TLS master secret for AUSF 103b.

According to the EAP-TLS authentication for RFC 5246, the derived TLS master secret can be input into a key derivation function such as described in section 2.3 titled “Key Hierarchy” of RFC 5246 in order to generate both a master session key (MSK) and an extended master session key (EMSK) 236b. For some exemplary embodiments, the extended master session key 236b can comprise a length of 512 bits. For exemplary embodiments, including those supporting 3GPP TS 33.501 versions 15.1.0, the first 256 bits of the EMSK 236b can be taken as the value for K_{ausf} in a step 236 in order to select an anchor key 236a which could comprise K_{ausf}. Note that (i) future versions of EAP-TLS authentication that explicitly support TLS version 1.3 can also derive a value of EMSK 236b from the TLS master secret as described in section 7.1 of RFC 8446, and (ii) future versions of 3GPP TS 33.501 can specify the derivation of K_{ausf} from the EMSK 236b.

In exemplary embodiments, a step 236 can comprise (i) AUSF 103b mutually deriving a TLS master secret according to RFC 8446 with device 101, and then (ii) AUSF 103b mutually deriving with device 101 an anchor key such as K_{ausf} using the mutually derived TLS master secret. An intermediate step between (i) and (ii) for both AUSF 103b and device 101 can be deriving an EMSK 236b according to IETF EAP-TLS standards, where the K_{ausf} is derived from the EMSK 236b. Other possibilities for AUSF 103b to mutually derive a shared secret key (different than key 225a, but using at least key 225a) in a step 236 using data from a TLS session such as the keys and values within messages 218, 227, and/or 223 are possible as well without departing from the scope of the present invention.

After a step 236, AUSF 103b can send MNO 102 and SEAF 102b a message 237 through secure session 201, where message 237 can include both the mutually derived (between AUSF 103b and device 101) anchor key 236a, which could comprise a K_{ausf}, and the value of SUPI 101a. Note that anchor key 236a can comprise a derived shared secret key that is mutually derived between AUSF 103b and device 101, and a different name than an “anchor key” of “K_{asuf}” could be used to describe the key 236a, such as a shared secret, a derived shared secret, a shared secret key, etc. Or, in some exemplary embodiments an AUSF 103b could send the TLS master secret or EMSK 236b in a message 237, and an SEAF 102b could derive anchor key 236a from the TLS master secret or EMSK 236b received in a message 237. The SEAF 102b can receive message 237 and conduct a step 238 in order to process the message. In a step 238, SEAF 102b can conduct additional key derivations using the anchor key 236a such as K_{asuf} in order to derive an associated key set 238a K_{seaf}, K_{amf}, K_{gnb}, etc., as specified in 3GPP TS 33.501.

As depicted in Figure 2, device 101 in a step 239 can conduct the equivalent steps as AUSF 103b in a step 236 in order to mutually derive a shared secret key such as an anchor key 236a or K_{ausf} using at least the TLS master secret. Device 101 can use the steps specified in section 7.1 of TLS version 1.3 RFC 8446 in order to derive the TLS master secret. Data input into the derivation of a TLS master secret for device 101 can comprise the shared secret 225a derived in a step 228 and the plaintext values for data transmitted in message 218 and 227 (e.g. after decryption of a message 227). The shared

secret 225a derived in a step 228 and the plaintext values for data transmitted in message 218 and 227 can be input into a key derivation function in order to derive a TLS master secret for device 101. The key derivation function can comprise a secure hash algorithm. Device 101 could take additional steps equivalent to those described in a step 236 above for an AUSF 103b to derive an EMSK 236b using at least derived TLS master secret. Corresponding to a step 236 for AUSF 103b above, a device 101 could derive an anchor key 236a from the EMSK 236b using standards specified for EAP-TLS authentication and/or 3GPP standards for 5G networks. A step 239 for device 101 can also comprise device 101 deriving an associated key set 239a using a derived Kausf from the TLS master secret for communication with MNO 102, such as a Kseaf, Kamf, Kgnb, etc. as specified in 3GPP TS 33.501.

After (i) a step 239 by device 101 to derive a shared secret key such as Kausf and related keys such as those specified in 3GPP TS 33.501 and (ii) a step 238 by MNO 102 with SEAF 102b to also use the shared secret such as Kausf received in a message 227 to derive related keys such as those specified in 3GPP TS 33.501, then device 101 and MNO 102 can exchange encrypted and integrity protected messages in RF communications 240. RF communications 240 can be conducted with keys 239a derived by device 101 in a step 239 and keys 238a derived from an anchor key by SEAF 102b in a step 238.

The steps for encrypting messages and also conducting integrity checks of messages in communications 240 can also be specified in 3GPP TS 33.501 and related standards. In addition, for embodiments where MNO 102 and device 101 use an 802.11 network instead of a 5G network, then device 101 and MNO 102 could use at least a portion of an EMSK 236b (derived from TLS master secret) in order to derive related encryption and integrity protection keys such as, but not limited to a pairwise transit key (PTK) in order to support the 802.11 network between device 101 and MNO 102. In summary, the encryption and integrity protection of messages in communications 240 are with keys derived from the TLS master secret, where the derivation of the TLS master secret is depicted and described above in a step 236 for AUSF 103b and a step 239 for device 101. In addition, the encryption and integrity protection of messages in communications 240 are with keys derived from shared secret 225a in a step 225 by AUSF 103b and the same shared secret 225a in a step 228 by device 101, since the TLS master secret for both nodes uses the shared secret 225a and data available to both device 101 and AUSF 103b such as the plaintext data in messages 218 and 227.

Note that the message 218 can include a random number generated by device 101 and the message 227 can include a random number generated by AUSF 103b, where the random numbers are also used in the derivation of the TLS master secret. In this manner, and by (i) including the random number generated by device 101 in a message 218 and the random number generated by AUSF 103b in a message 227, and (ii) by including the data from a message 218 and a message 227 in the derivation of a TLS master secret, additional entropy or random values can be used by AUSF 103b and device 101 in addition to mutually derived shared secret 225a for the derivation of a TLS master secret. As

described above, the TLS master secret can be used to derive a shared secret or an anchor key 236a for device 101 and AUSF 103b and MNO 102 to use in subsequent communications such as a connection for messages 240. In other words, at least a first random number within a “client hello” message 218 and a second random number within a “server hello” message 227, along with the shared secret 225a can be used by device 101 and AUSF 103b to derive a shared secret for subsequent communications 240, where the shared secret can comprise an anchor key 236a. For 802.11 networks between device 101 and MNO 102, the shared secret can comprise at least a portion of EMSK 236b.

For some exemplary embodiments, the wireless networking technology supported by MNO 102 can comprise WiFi networking supporting IEEE 802.11 standards. For these embodiments, the anchor key 236a sent in a message 237 through secure session 201 to MNO 102 can comprise any of (i) the shared secret key 225a, (ii) the TLS master secret derived in a step 236, and/or (iii) the value of EMSK 236b. In other words, when supporting WiFi networking for EAP-TLS authentication for device 101 and MNO 102, network 103 could send a mutually derived anchor key 236a that is different than a Kasuf as contemplated by 5G standards in 3GPP TS 33.501. Other possibilities exist as well for the structure or contents of a mutually derived anchor key 236a sent in a message 237 from network 103 to wireless network 102 without departing from the scope of the present invention. Further, when supporting 802.11 wireless networks for device 101 and MNO 101, the use of a SUCI according to 3GPP TS 33.501 can be optional, and the device identity as received by network 103 in a message 205 can comprise a SUPI 101a without encryption and also without the device ephemeral public key ePK.ECIES.Device 101h.

Figure 3

Figure 3 is an illustration of a device database and a network database with exemplary data stored, in accordance with exemplary embodiments. A device provider 104 could store and operate a device database 104d as depicted and described in connection with Figure 1. Network 103 could store and operate a network database 103d in connection with Figure 1. The device database 104d or network database 103d could be recorded in a server 111 or collection of servers securely connected to device provider 104 or network 103, respectively. Each database could operate using software such as Oracle, Microsoft SQL Server, MySQL, and other possibilities exist for a database without departing from the scope of the present disclosure.

Although both a device database 104d and network database 103d depict multiple values, strings, or numbers stored in each column and row, some values could be omitted at certain points in time, such as the data not being available at a point in time, although the omitted data could be provided or available later. For example, with a network database 103d, upon receipt of a message 205 with a new SUCI 101b not previously stored in a network database 103d, network 103 could insert a new row into a network database 103d with SUCI 101b entered and the other data empty, until later received in

later messages (i) such as message 213 from device provider 104 or (ii) received from device 101 via MNO 102. Both device database 104d and network database 103d can be encrypted with a symmetric ciphering key such that the data is stored in physical memory such as storage 111b as ciphertext.

Values depicted in Figure 3 with different numeral designations such as “-1”, “-2”, “-3”, etc. can represent different numbers or strings for the depicted value such as SUPI 101a. Values depicted in Figure 3 with the same numeral designation such as the first three rows of Provider ID 104a in a device database 104d being “104a-1”, then “104a-1”, then “104a-1” can represent that the number or string for provider ID 104a is the same for each of the devices in the first three rows. Values depicted in Figure 3 with different numeral designation such as the last three rows of Provider ID 104a in a device database 104d being “104a-2”, then “104a-2”, then “104a-2” can represent that the number or string for provider ID 104a is different than each of the devices in the first three rows, which are depicted with a designation being “104a-1”.

Device database 104d can record data for a plurality of devices 101 which could be managed or supported by device provider 104. A device database 104 could store values, numbers, or strings for a device number or index, a user name for a network access identifier (NAI), a device identity as a SUPI 101a, a device certificate 101c, a provider identity 104a, a provider certificate 104c, a provider secret key 104e, ECIES secret key 104b, and ECIES parameters 101f. Data within a device database 104d could be recorded or stored before device 101 establishes an initial radio resource connection with MNO 102, such as during device manufacturing or distribution or a device configuration, and other possibilities exist as well for the time before a message 205 when data for device database 104d is recorded. As additional devices are added to device provider 104, then additional rows for the additional devices could be inserted into a device database 104d.

As depicted in Figure 3, a device database could store additional data for a device 101 than the fields or values depicted. Additional data could include the ECIES ephemeral public key 101h received in a message 209 by device provider 104, the SUCI 101b used by a device 101 or a device user name associated with a device 101. In addition, although not depicted for a device database 104d in Figure 3, a device database 103d could also store a first list of cryptographic algorithms 101s and a second list of cryptographic parameters 101x for a device 101 with SUPI 101a in a device database 104d, where the first list and the second list could be used to determine extensions 219d from an AUSF 103b are supported by device 101. The first list and the second list could also be sent to network 103 in message 213 in Figure 2 above. Other possibilities exist as well for data stored or recorded in a device database 104d without departing from the scope of the present disclosure.

Although a single device database 104d is depicted in Figure 3, a device database 104d could comprise either multiple databases or multiple tables with data equivalent or similar to that depicted in Figure 3. In one exemplary embodiment, a device provider 104 could operate multiple device databases 104d, where a first device database 104d is associated with a first network 103 and a second device

database 104d is associated with a second network 103, etc. The realm values for a NAI within a SUCI or SUPI could include a first value for the network portion and/or provider portion of the realm for the first network and first device database. The realm values for a NAI within a SUCI or SUPI could include a second value for the network portion and/or provider portion of the realm for the second network and second device database.

The user name in a device database can comprise the user name portion of a network access identifier (NAI) for a SUPI 101a. The SUPI 101a stored in a device database 104d can comprise the SUPI 101a as described above for a device 101 in Figure 1. Note that the user name can be plaintext with a SUPI. In addition, a “user name” can comprise an identity for a device 101 and a “user name” does not need to be associated with a person. Device certificate 101c stored in a device database 101d can comprise the device certificate 101c stored and used by device 101 as described for a device 101 in Figure 1. The format and data structure of device certificate 101c in a network database 103d, device provider database 104d, and device 101 could take many forms such as a *.pem structure, raw text, *.crt, or *.der. *.crt can comprise a certificate format and *.der can comprise distinguished encoding rules for the certificate. Other certificates described herein, including device provider certificate 104c could be stored or transmitted in any of the exemplary formats described in the previous sentence.

The provider ID 104a stored in a device database 104d can comprise the provider identity used in the realm portion of a SUCI 101b and SUPI 101a for a device 101. The provider ID 104a in a device database 104d can also comprise the identity for a device provider stored in a provider certificate 104c, such as an identity in the “common name” field or also possibly a portion of the “subject alternative name”, and other possibilities exist as well for the location of the provider ID 104a within provider certificate 104c. Note that the provider ID 104a in a SUCI 101b or SUPI 101a is not required to equal or be the same as a provider ID 104a in the provider certificate 104c. A database 104d can store the provider ID 104a in the provider certificate 104c and also the different provider ID 104a in the SUPI 101a. Provider certificate 104c can be stored in device database 104d, network database 103d, and device 101 in a format or structure similar to device certificate 101c described in the paragraph above. In some exemplary embodiments provider certificate 104c can comprise a certificate in *.pem format.

The device provider certificate 104c in a device database 104d and network database 103d can comprise the device provider certificate 104c depicted and described in Figure 1 and Figure 2. As depicted in Figure 3, different devices 101 can use different provider certificates 104c, which is also depicted in device database 104d and network database 103d. Note that the same provider ID 104a can be used with different provider certificates 104c, such as provider ID “104a-2” being associated with both provider certificate “104c-2” and provider certificate “104c-3”. For this embodiment, the same provider ID 104a could be used with two different certificates with different parameters such as different certificate expiration dates or different key lengths or key names or different elliptic curves used for the public keys, etc., such that different devices 101 could use different provider certificates 104c. In

addition, a device 101 using a SUPI 101a could store or be associated with a plurality of different device provider certificates 104c, and the multiple different device provider certificates 104c could be stored for the device 101 in a device database 104d. A specific device provider certificate 104c for the device 101 could be selected based on the network 103 or MNO 102 communicating with device 101, such as parameters supported by the network 103 or MNO 102, where the parameters could be received by device provider 104 via secure session 202.

In exemplary embodiments, the server certificate for a network 103 when conducting an EAP-TLS authentication with device 101 can use the device provider certificate 104c, where the different server certificates used by network 103 and AUSF 103b are selected in a step 219b and identified based on the provider ID 104a in the realm of the SUPI 101a or SUCI 101b. Also note that a device database 104d and a network database 103d could record unique certificate identities for the depicted certificates, and the full certificates as exemplary *.pem files could be stored outside of a database (but identified by the unique certificate identities).

The provider secret key 104e for a provider certificate 104c can be used with the provider public key stored with the device provider certificate 104c. The use of a provider secret key 104e was described above in Figure 1 and Figure 2, such as selecting in a step 222 the provider secret key 104e from a device database 104d based on data in a message 220 such as a device provider certificate 104c for the “message to sign”. Device provider 104 could use the selected device provider secret key 104e in order to conduct the step 222 to create or generate the “Certificate Verify” 222a portion or segment of the TLS “server hello” message subsequent sent by AUSF 103b in a message 227 in Figure 2.

Device database 104d can also store values of ECIES secret key 104b, which corresponds to the ECIES public key 101e stored by device 101. Note that in some exemplary embodiments a device provider 104 can select the ECIES secret key 104b based on the provider ID 104a received within a realm value for the SUCI 101b from a message 209 as depicted in Figure 2. In other words, in exemplary embodiments, there can be a one-to-one mapping or a unique mapping between ECIES secret key 104b and the provider ID 104a in a realm for a device provider 104 in a SUCI 101b from message 209 using a device database 104d. In this manner, a device provider can know which ECIES secret key 104b to use for a plurality of devices 101, where the ECIES secret key 104b to use with an ECIES decryption function 104j can be identified based on the provider ID 104a received in a realm of the SUCI 101b.

Although exemplary data is depicted within a single table within device database 104d in Figure 3, a device database 104d could include multiple different tables. As one example, a separate table could include a “one-to-one” mapping between a provider ID 104a, which could be included in the realm portion of a SUCI 101b, and the value ECIES secret key 104b and the value of ECIES parameters 101f. Note that a device database 104d could also store the of ECIES parameters 101f in a device database 104d, which can specify parameters for use with the ECIES secret key 104b, such as use of

curve 25519 for a first ECIES secret key 104b and the use of curve P-256 for a second ECIES secret key 104b, and other possibilities exist as well.

As depicted in Figure 1, a network 103 could store and operate a network database 103d. Either device database 104d or network database 103d could comprise a collection of servers and storage memory for storing values associated with a plurality of devices in order to support the operation of a system 200 or a system 100. A network database 103d could store values, strings, or numbers for Device identity / SUCI 101b, Device identity / SUPI 101a, Device Certificate 101c, Provider Certificate 104c, TLS Server Ephemeral secret key 103q, TLS Device Ephemeral public key 101n, Shared Secret 225a, Symmetric key 225b, EMSK 236b, and Anchor Key 236a. Additional supporting data for a system 200 could be stored or recorded in a network database 103d as well without departing from the scope of the present disclosure.

Data within a network database 103d could be recorded or stored as network 103 receives data for a device 101 from a MNO 102 and a device provider 104. For example, as a network 103 receives a SUCI 101b for a device 101 from a MNO 102 after device 101 established a radio resource connection with MNO 102 and MNO 102 forwards the SUCI 101b in a message 205 in Figure 2, then the value for SUCI 101b could be added to the network database 103d. As additional data from additional devices 101 is received by network 103, then additional rows for the additional devices could be inserted into a network database 103d. A network database could store additional data or columns for a device 101 than the fields or values depicted.

Additional data in a network database 103d could include the ECIES ephemeral public key 101e received in a message 205 by network 103, and an identity of MNO 102 of MNO ID 102a that sent a message 205. For embodiments where network 103 uses a new server certificate instead of a device provider certificate 104c in the “server hello message” (as described above for an alternative embodiment of message 220 with a “certificate signing request”), then the new server certificate or an identity of the new server certificate could be stored for a device 101 along with the SUCI 101b. Additional data in a network database 103d could include the selected extensions 219d used for generating a “server hello” in message 227, where the extensions 219d could be selected in a step 219c in Figure 2 above and other possibilities exist as well without departing from the scope of the present disclosure.

Although a single network database 103d is depicted in Figure 3, a network database 103d could comprise either multiple databases or multiple tables with data equivalent or similar to that depicted in Figure 3. In one exemplary embodiment, a network 103 could operate multiple network databases 103d, where a first network database 103d is associated with a first device provider 104 and a second network database 104d is associated with a second device provider 104, etc. The realm values for a NAI within a SUCI or SUPI could include a first value for the device provider portion the realm for the first device provider and first device database. The realm values for a NAI within a SUCI or SUPI could include a

second value for the device provider portion of the realm for the second device provider and second device database.

Values for Device identity / SUCI 101b in a network database 103d could be received in a message 205 from MNO 102 in Figure 2. Values in a network database 103d of Device identity / SUPI 101a, Device Certificate 101c, and Provider Certificate 104c could be received in a message 213 in Figure 2. Values for TLS Server Ephemeral secret key 103q could be derived in a step 103q' as depicted and described in connection with a step 103q' in Figure 2. Values for TLS Device Ephemeral public key 101n in a network database 103d could be received in a message 218 from Figure 2. Values for shared secret key 225a and symmetric ciphering key 225b could be derived in a step 225 in Figure 2. Values for EMSK 236b and an anchor key 236a or Kausf could be derived in a step 236 in Figure 2.

Network 103 can use the data recorded in a network database 103d in order to conduct the series of steps and process the series of messages as depicted and described in connection with Figure 2. As one example and as described for a step 219b, network 103 and/or AUSF 103b could use data recorded in a network database 103d with the SUCI 101b value from a message 218a to select at least a device provider certificate 104c as the server certificate to use with EAP-TLS authentication and the generation of a "server hello" in message 227. The values for TLS Server Ephemeral secret key 103q and TLS Device Ephemeral public key 101n, along with associated parameters could be selected in a step 225 using the SUCI 101b from a message 218a in order to conduct a step 225 to generate the shared secret 225a and the symmetric ciphering key 225b. The ciphertext in a "server hello" within message 227 can be encrypted with the symmetric ciphering key 225b and the ciphertext in a message 233 can be decrypted with the symmetric ciphering key 225b from a network database 103d. The message 233 can be received from MNO 102 with a SUCI 101b, and the symmetric ciphering key 225b selected for decryption from the network database 103d using the SUCI 101b.

The received device certificate 101c decrypted from a message 233 using the symmetric ciphering key 225b can be compared with the device certificate 101c stored in a database for SUCI 101b using a step 224. Network 103 can derive an anchor key 236a for a device 101 with a SUCI 101b using at least the shared secret 225a and the EMSK 236b stored in a network database 103d for the SUCI 101b. Additional examples for the use of data in a network database 103d were provided in Figure 2 above, and other possibilities exist as well for a network 102 and/or an AUSF 102b to store and use data from a network database 103d without departing from the scope of the present disclosure.

Conclusion

Various exemplary embodiments have been described above. Those skilled in the art will understand, however, that changes and modifications may be made to those examples without departing from the scope of the claims.

CLAIMS

1. A method for a server system to support an EAP-TLS authentication with a device, the method performed by the server system, the method comprising:

- a) receiving, from a wireless network, a subscriber concealed identifier (SUCI) and a first device ephemeral public key;
- b) sending, to a device provider, the SUCI and the first device ephemeral public key;
- c) receiving, from the device provider, a subscriber permanent identifier (SUPI), a certificate for the device, and a device provider certificate;
- d) receiving, from the wireless network, a “client hello” message for the device with at least a second device ephemeral public key;
- e) deriving a server ephemeral private key and a corresponding server ephemeral public key;
- f) sending, to the device provider, at least the derived server ephemeral public key;
- g) receiving, from the device provider, a digital signature over at least the derived server ephemeral public key and the device provider certificate, wherein the digital signature is generated by the device provider using at least a device provider private key corresponding to a device provider public key in the device provider certificate;
- h) generating (i) a shared secret key using at least the second device ephemeral public key and the server ephemeral private key and (ii) a shared symmetric ciphering key using at least the shared secret key;
- i) sending, to the wireless network, the SUCI, a “server hello” message with at least the derived server ephemeral public key, the device provider certificate, and the digital signature, wherein at least the device provider certificate and the digital signature in the “server hello” message are encrypted with the shared symmetric ciphering key; and
- j) generating an anchor key for the wireless network using at least the shared secret key, the “client hello” message, and the “server hello” message.

2. The method of claim 1, wherein the “client hello” message and the “server hello” message support a Transport Layer Security version 1.3, and wherein the device generates the “client hello” message.

3. The method of claim 1, wherein the SUCI includes a domain name for the device provider as a first plaintext and a user name for the device as a ciphertext, and wherein the SUPI includes the user name for the device as a second plaintext.

4. The method of claim 1, wherein the “client hello” message includes extensions, and wherein the extensions include cryptographic parameters, and wherein the server ephemeral private key, the corresponding server ephemeral public key, and the second device ephemeral public key support the cryptographic parameters.

5. The method of claim 4, further comprising generating, by the server system, the shared secret using an elliptic curve Diffie Hellman (ECDH) key exchange and the cryptographic parameters.

6. The method of claim 1, wherein the server system comprises at least one server and an authentication server function (AUSF).

7. The method of claim 1, wherein device provider records a secret key, and wherein the device provider decrypts at least a portion of the SUCI using the secret key and the first device ephemeral public key in order to determine the SUPI for the device.

8. The method of claim 1, wherein the device stores the device provider certificate before the device sends the SUCI to the wireless network, and wherein the device verifies the digital signature using at least the device provider public key in the device provider certificate.

9. The method of claim 1, wherein the wireless network encrypts data for the device using at least the anchor key.

10. The method of claim 1, further comprising generating the shared symmetric ciphering key using at least the shared secret key and a key derivation function with the shared secret key, and wherein the key derivation function uses a secure hash algorithm.

11. The method of claim 1, further comprising, after step i) and before step j), receiving from the device a “client finished” message with a certificate verify for the device, wherein the server system verifies the certificate verify using the device certificate.

12. The method of claim 11, further comprising generating the anchor key for the wireless network using at least the shared secret key, the “client hello” message, the “server hello” message, and the “client finished” message.

13. The method of claim 1, wherein the server system sends the wireless network the anchor key through a secure session established between the server system and the wireless network.

14. The method of claim 1, wherein the SUCI includes a plaintext realm for the device provider, and wherein the server system uses the plaintext realm to send the SUCI and the first device ephemeral public key to the device provider.

15. A method for a server system to support an EAP-TLS authentication with a device, the method performed by the server system, the method comprising:

- a) storing a static private key and cryptographic parameters, wherein the device stores a static public key corresponding to the static private key and the cryptographic parameters;
- b) receiving, from a wireless network, a subscriber concealed identifier (SUCI) and a first device ephemeral public key for the cryptographic parameters, wherein the SUCI includes a realm for a device provider as a first plaintext;
- c) decrypting the SUCI using at least the static private key and the first device ephemeral public key in order to read a subscriber permanent identifier (SUPI);
- d) sending, to a device provider at the realm, at least a user name for the SUPI;
- e) receiving, from the device provider, a certificate for the device and a device provider certificate;
- f) receiving, from the wireless network, a “client hello” message for the device with at least a second device ephemeral public key;
- g) deriving a server ephemeral private key and a corresponding server ephemeral public key;
- h) sending, to the device provider, at least the derived server ephemeral public key;
- i) receiving, from the device provider, a digital signature over at least the derived server ephemeral public key and the device provider certificate, wherein the digital signature is generated by the device provider using at least a device provider private key corresponding to a device provider public key in the device provider certificate;
- j) generating (i) a shared secret key using at least the second device ephemeral public key and the server ephemeral private key and (ii) a shared symmetric ciphering key using at least the shared secret key;
- k) sending, to the wireless network, the SUCI, a “server hello” message with at least the derived server ephemeral public key, the device provider certificate, and the digital signature, wherein at least the device provider certificate and the digital signature in the “server hello” message are encrypted with the shared symmetric ciphering key; and
- l) generating an anchor key for the wireless network using at least the shared secret key, the “client hello” message, and the “server hello” message.

16. The method of claim 15, wherein the “client hello” message and the “server hello” message support a Transport Layer Security version 1.3, and wherein the device generates the “client hello” message.

17. The method of claim 15, wherein the SUCI includes a user name for the device as a ciphertext, and wherein the SUPI includes the user name for the device as a second plaintext.

18. The method of claim 15, wherein the “client hello” message includes extensions, and wherein the extensions include cryptographic parameters, and wherein the server ephemeral private key, the corresponding server ephemeral public key, and the second device ephemeral public key support the cryptographic parameters.

19. The method of claim 18, further comprising generating, by the server system, the shared secret using an elliptic curve Diffie Hellman (ECDH) key exchange and the cryptographic parameters.

20. The method of claim 15, wherein the server system comprises at least one server and an authentication server function (AUSF).

21. The method of claim 15, wherein device provider stores a device database for a plurality of devices, and wherein the device provider queries the device database with the user name in order to send the server system the certificate for the device and the device provider certificate.

22. The method of claim 15, wherein the device stores the device provider certificate before the device sends the SUCI to the wireless network, and wherein the device verifies the digital signature using at least the device provider public key in the device provider certificate.

23. The method of claim 15, wherein the wireless network encrypts data for the device using at least the anchor key.

24. The method of claim 15, further comprising generating the shared symmetric ciphering key using at least the shared secret key and a key derivation function with the shared secret key, and wherein the key derivation function uses a secure hash algorithm.

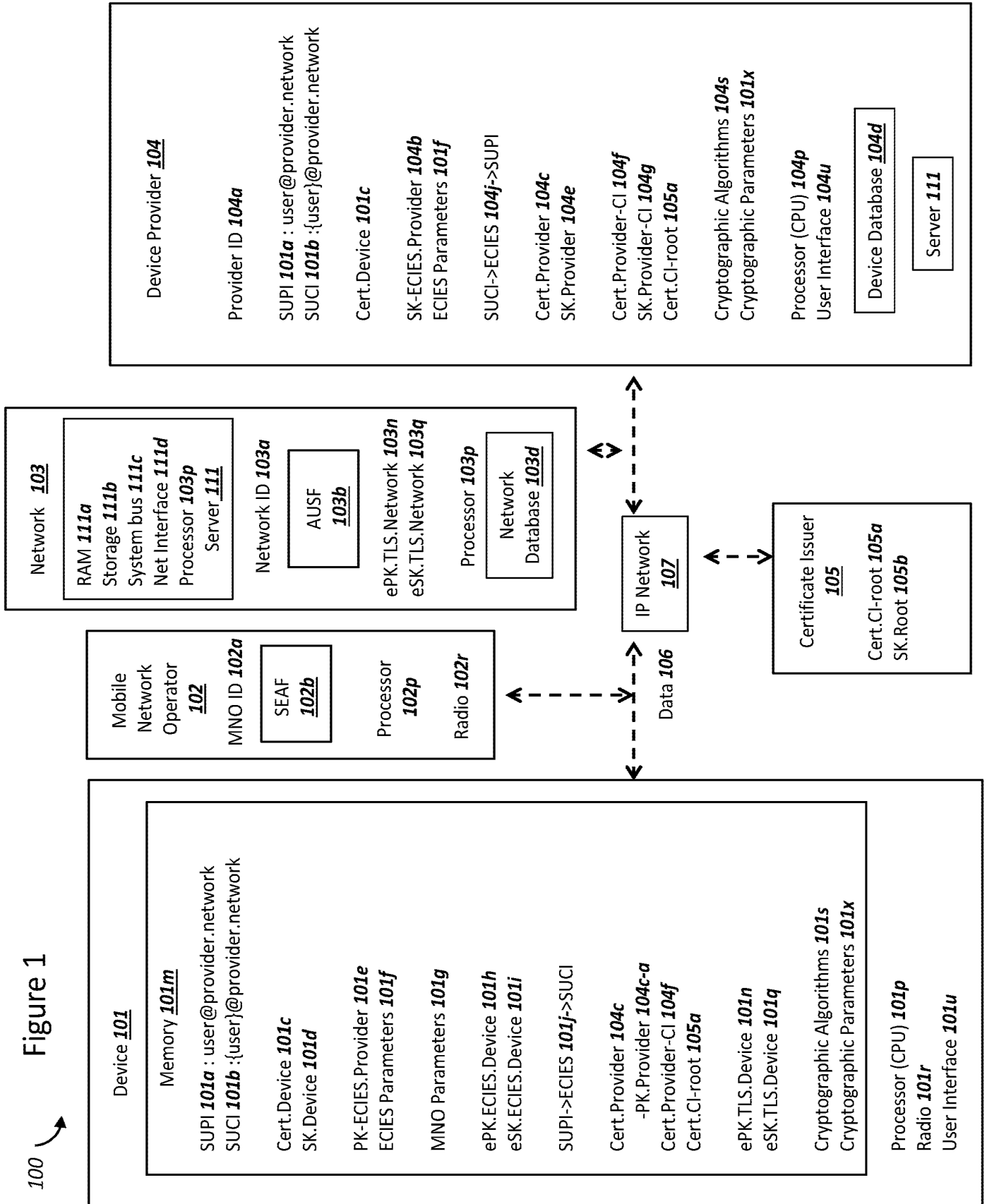
25. The method of claim 15, further comprising, after step i) and before step j), receiving from the device a “client finished” message with a certificate verify for the device, wherein the server system verifies the certificate verify using the device certificate.

26. The method of claim 25, further comprising generating the anchor key for the wireless network using at least the shared secret key, the “client hello” message, the “server hello” message, and the “client finished” message.

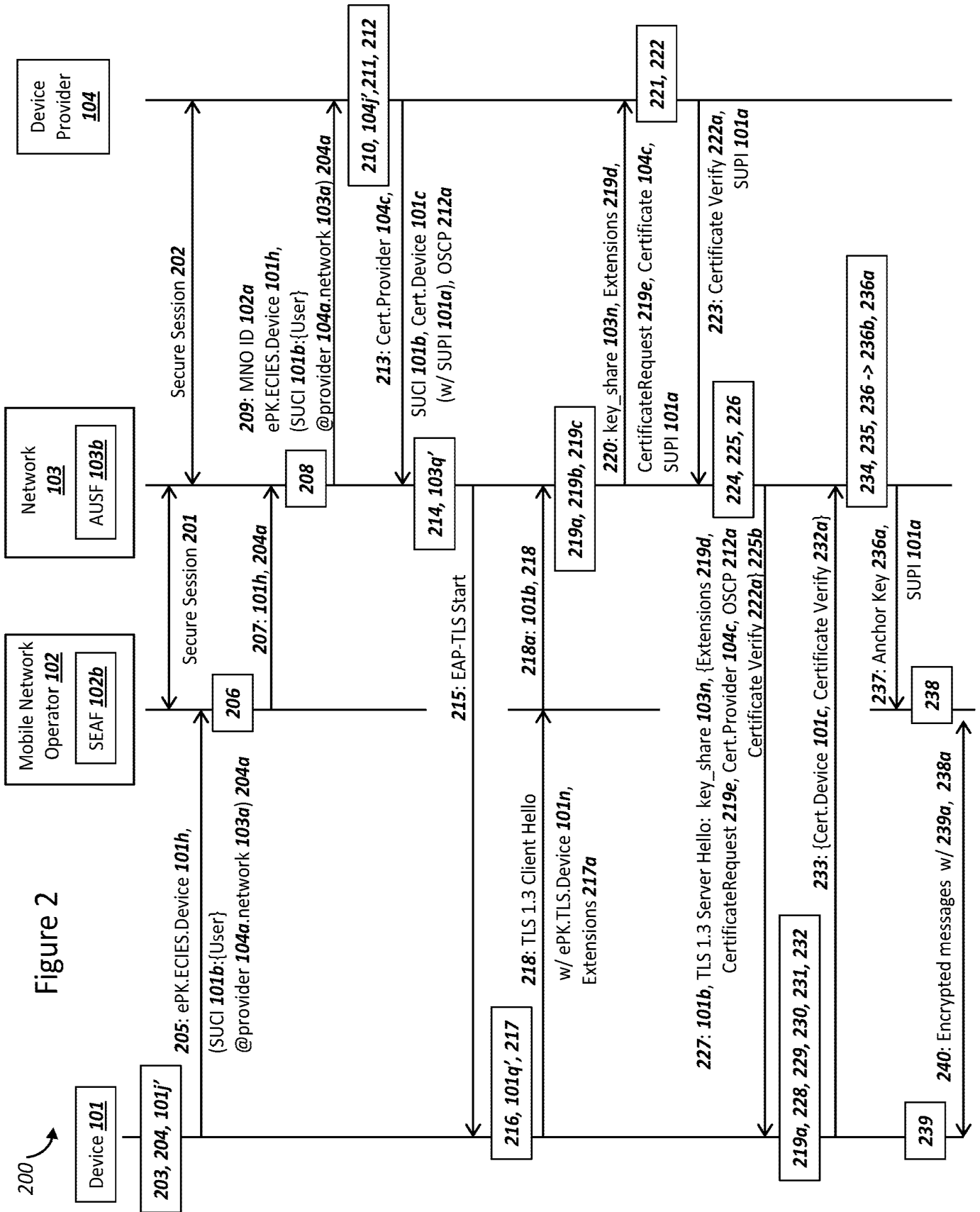
27. The method of claim 15, wherein the server system sends the wireless network the anchor key through a secure session established between the server system and the wireless network.

28. The method of claim 15, wherein the device stores an first device ephemeral private key for the first device ephemeral public key, and wherein the device encrypts the SUPI using the static public key and the first device ephemeral private key in order to generate the SUCI, and wherein the device sends the SUCI to the wireless network.

1/3



2/3



3/3

Figure 3

Device Database <u>104d</u>									
Device	NAI User		Device ID /		Device	Provider	Provider	Provider	ECIES Secret Key
	Name	SUPI	Certificate	ID	Certificate	Secret Key	ECIES Secret Key	ECIES Parameters	
		101a	101c	104a	104c	104e	104b	101f	• • •
1	user-1	101a-1	101c-1	104a-1	104c-1	104e-1	104b-1	101f-1	
2	user-2	101a-2	101c-2	104a-1	104c-1	104e-1	104b-1	101f-1	
3	user-3	101a-3	101c-3	104a-1	104c-2	104e-2	104b-1	101f-1	
4	user-4	101a-4	101c-4	104a-2	104c-2	104e-2	104b-2	101f-2	
5	user-5	101a-5	101c-5	104a-2	104c-3	104e-3	104b-2	101f-2	
6	user-6	101a-6	101c-6	104a-2	104c-3	104e-3	104b-2	101f-2	
				• • •					
Network Database <u>103d</u>									
Device ID / SUPI	Device ID /		Device	Provider	TLS Server	TLS Device	Shared	Symmetric	Anchor Key
	SUPI	Certificate	Certificate	Ephemeral SK	Ephemeral PK	Secret	key	EMSK	
101b	101a	101c	104c	103q	101n	225a	225b	236b	236a
101b-1	101a-1	101c-1	104c-1	103q-1	101n-1	225a-1	225b-1	206b-1	206a-1
101b-2	101a-2	101c-2	104c-1	103q-2	101n-2	225a-2	225b-2	206b-2	206a-2
101b-3	101a-3	101c-3	104c-2	103q-3	101n-3	225a-3	225b-3	206b-3	206a-3
101b-4	101a-4	101c-4	104c-2	103q-4	101n-4	225a-4	225b-4	206b-4	206a-4
101b-5	101a-5	101c-5	104c-3	103q-5	101n-5	225a-5	225b-5	206b-5	206a-5
101b-6	101a-6	101c-6	104c-3	103q-6	101n-6	225a-6	225b-6	206b-6	206a-6
				• • •					

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 2020/030424

A. CLASSIFICATION OF SUBJECT MATTER

H04W 12/06(2009.01)**H04W 12/04**(2009.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04W 12/00-12/12, H04W80/00, H04W84/00, H04L9/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Espacenet, PatSearch, PAJ, WIPO, USPTO

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 2019/020439 A1 (TELEFONAKTIEBOLAGET LM ERICSSON) 31.01.2019, abstract, [0016], [0017]-[0020], [0046], [0048], [0049], [0076], [0081]), fig. 3, 6	1-28
Y	WO 2018/208228A2 (HUAWEI INT PTE LTD) 15.11.2018, [0137], [0336]-[0363], fig. 15	1-28
Y	5G; Security architecture and procedures for 5G System (3GPP TS 33.501 version 15.1.0 Release 15) [online]. July 2018 [retrieved 04.07.2020]. Retrieved from < https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.01.00_60/ts_133501v150100p.pdf >, chapter 6.1.1	15-28



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“D” document cited by the applicant in the international application

“E” earlier document but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T”

later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X”

document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y”

document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&”

document member of the same patent family

Date of the actual completion of the international search

07 July 2020 (07.07.2020)

Date of mailing of the international search report

23 July 2020 (23.07.2020)

Name and mailing address of the ISA/RU:

Federal Institute of Industrial Property,
Berezhkovskaya nab., 30-1, Moscow, G-59,
GSP-3, Russia, 125993
Facsimile No: (8-495) 531-63-18, (8-499) 243-33-37

Authorized officer

E. Krivtsova

Telephone No. 8(499)240-25-91

Form PCT/ISA/210 (second sheet) (July 2019)

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 2020/030424

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	Badra M. Key-Exchange Authentication Using Shared Secrets. Computer, 39(3) [online]. March 2006, [retrieved 04.07.2020]. Retrieved from < https://www.researchgate.net/publication/2956506_Key-Exchange_Authentication_Using_Shared_Secrets >, pages 58-66, chapters «Certificate issues in wireless networks», «Transport Layer Security-Key-Exchange Method Benefits», «The Transport Layer Security Key-Exchange Method (TLS-KEM) extension uses an out-ofband shared secret to integrate TLS with wired and wireless networks without influencing security robustness», «Discussion and results»	15-28
Y	E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3 [online]. August 2018 [retrieved 04.07.2020]. Retrieved from < https://tools.ietf.org/html/8446 >, chapters 2, 4.1.2, 4.1.3, 4.1.6, 4.2.3, 4.2.4, 4.4.1, 4.4.2, 7.3 7.4.2, 8.1	2, 4, 5, 8-13, 16, 18-27
A	US 7774411 B2 (WISYS TECHNOLOGY FOUNDATION, INC.) 10.08.2010	1-28

Form PCT/ISA/210 (continuation of second sheet) (July 2019)



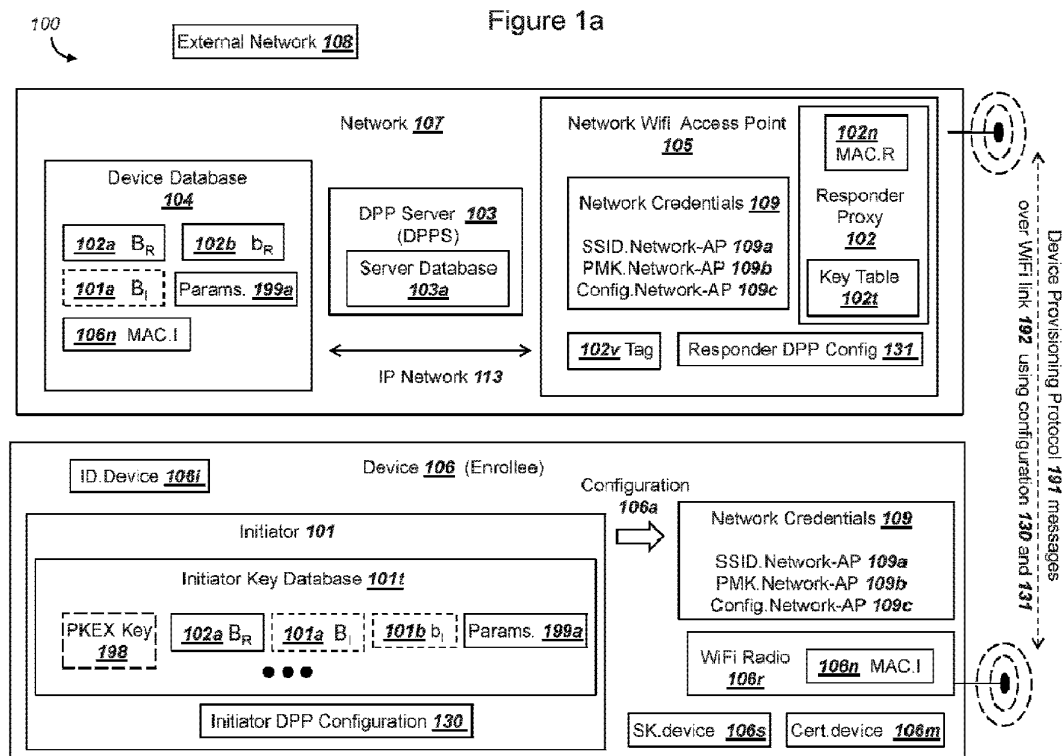
(12) **DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION**

(13) **A1**

(86) Date de dépôt PCT/PCT Filing Date: 2019/05/15
(87) Date publication PCT/PCT Publication Date: 2019/11/21
(85) Entrée phase nationale/National Entry: 2021/02/23
(86) N° demande PCT/PCT Application No.: US 2019/032371
(87) N° publication PCT/PCT Publication No.: 2019/222319
(30) Priorité/Priority: 2018/05/17 (US62/672,977)

(51) Cl.Int./Int.Cl. *G06F 21/30* (2013.01),
G06F 21/62 (2013.01)
(71) Demandeur/Applicant:
IOT AND M2M TECHNOLOGIES, LLC, US
(72) Inventeur/Inventor:
NIX, JOHN A., US
(74) Agent: HICKS & ASSOCIATES

(54) Titre : PROTOCOLE HEBERGE DYNAMIQUE DE FOURNITURE DE SERVEURS ET UN ACCEPTEUR MIS EN RESEAU
(54) Title: A HOSTED DYNAMIC PROVISIONING PROTOCOL WITH SERVERS AND A NETWORKED RESPONDER



(57) Abrégé/Abstract:

A network can operate a WiFi access point with credentials. An unconfigured device can support a Device Provisioning Protocol (DPP), and record bootstrap public keys and initiator private keys. The network can record bootstrap public and responder private

(57) **Abrégé(suite)/Abstract(continued):**

keys and operate a DPP server. A responder proxy can establish a secure and mutually authenticated connection with the network. The network can (i) derive responder ephemeral public and private keys, (ii) record the initiator bootstrap public key, and (iii) select a responder mode for the responder. The network can derive a shared secret with at least the (i) recorded the initiator bootstrap public key and (ii) derived responder ephemeral private key. The network can encrypt credentials using at least the derived shared secret and send the encrypted credentials through the responder proxy to the initiator, which can forward the encrypted credentials to the device, thereby supporting a device configuration.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property
Organization
International Bureau

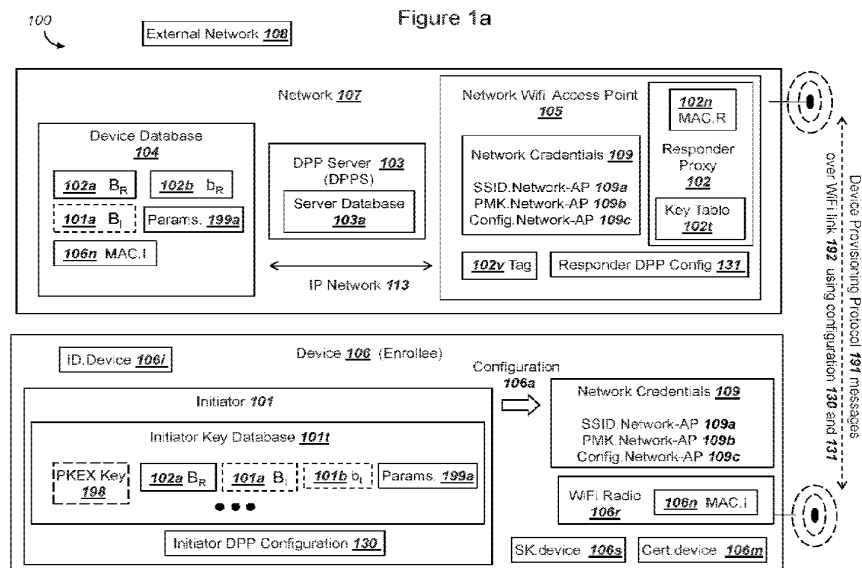
(43) International Publication Date
21 November 2019 (21.11.2019)



(10) International Publication Number
WO 2019/222319 A1

- (51) International Patent Classification:
G06F 21/30 (2013.01) *G06F 21/62* (2013.01)
- (21) International Application Number:
PCT/US2019/032371
- (22) International Filing Date:
15 May 2019 (15.05.2019)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
62/672,977 17 May 2018 (17.05.2018) US
- (71) Applicant: **IoT AND M2M TECHNOLOGIES, LLC**
[US/US]; 807 Davis Street, Unit 2207, Evanston, IL
60201-7104 (US).
- (72) Inventor: **NIX, John, A.**; c/o IoT And M2M Technologies, LLC, 807 Davis Street, Unit 2207, Evanston, IL
60201-7104 (US).
- (74) Agent: **THOMSON, Kirsten, L.**; McDonnell Boehnen
Hulbert & Berghoff LLP, 300 South Wacker Drive, Chicago,
IL 60606 (US).
- (81) Designated States (*unless otherwise indicated, for every
kind of national protection available*): AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ,
CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO,
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,
HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP,
KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME,
MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ,
OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA,

(54) Title: A HOSTED DYNAMIC PROVISIONING PROTOCOL WITH SERVERS AND A NETWORKED RESPONDER



(57) **Abstract:** A network can operate a WiFi access point with credentials. An unconfigured device can support a Device Provisioning Protocol (DPP), and record bootstrap public keys and initiator private keys. The network can record bootstrap public and responder private keys and operate a DPP server. A responder proxy can establish a secure and mutually authenticated connection with the network. The network can (i) derive responder ephemeral public and private keys, (ii) record the initiator bootstrap public key, and (iii) select a responder mode for the responder. The network can derive a shared secret with at least the (i) recorded the initiator bootstrap public key and (ii) derived responder ephemeral private key. The network can encrypt credentials using at least the derived shared secret and send the encrypted credentials through the responder proxy to the initiator, which can forward the encrypted credentials to the device, thereby supporting a device configuration.

[Continued on next page]

WO 2019/222319 A1

SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN,
TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

A HOSTED DYNAMIC PROVISIONING PROTOCOL WITH SERVERS AND A NETWORKED RESPONDER

CROSS-REFERENCE TO RELATED APPLICATION

5 This international PCT application claims the benefit of the filing date of U.S. Provisional Patent Application Serial No. 62/672,977, filed May 17, 2018, which is hereby incorporated by reference in its entirety.

BACKGROUND

Technical Field

10 The present systems and methods relate to authentication and configuration of WiFi radios, and more particularly, to using a Device Provisioning Protocol with a networked responder and a set of servers that record PKI keys, in order securely transfer a set of credentials to a device.

Description of Related Art

15 The ability to connect transducers such as sensors and actuators with a wireless network is a growing field with many economical applications. As the costs for both electronic hardware and bandwidth continue to decrease, the use of networked transducers is expected to continue increasing over the coming decades. Connecting transducers to a network can be referred to as “machine-to-machine (M2M) communications” or “the Internet of Things (IoT).” Among many potential benefits,
20 IoT technologies allow automated monitoring and/or control of assets, equipment, personnel, or a physical location where manual monitoring may not be economical. Many applications for the “Internet of Things” significantly reduce costs by using automated monitoring instead of manual techniques. Prominent examples of IoT applications today include monitoring and control for building heating/air-conditioning, automobiles, alarm systems, and tracking devices. Fast growing markets for
25 IoT applications today include health applications such as the remote monitoring of a person’s fitness activity, heartbeat, or glucose levels, monitoring of industrial equipment deployed remotely in the field, and also security systems.

Many IoT applications can leverage wireless networking technologies, in addition to wired technologies such as Ethernet. Wireless technologies such as wireless local area networks and wireless
30 wide area networks have proliferated around the world over the past 30 years, and usage of these wireless networks is also expected to continue to grow. Wireless local area network (LAN) technologies include WiFi based on the series of 802.11 standards from the Institute of Electrical and Electronics Engineers (IEEE). The many options to connect a transducer device to a wireless network

create opportunities for new products and services, but also create several classes of problems that need to be solved. Many of these problems or needs in the art pertain to securely and efficiently configuring the transducer devices. Without configuration including a set of network access credentials, a device cannot normally securely connect with a wireless network. A need exists in the art to allow a user or a network to securely and easily upload to the device a set of network access credentials for a wireless network.

Manually configuring devices to (i) use different security keys, (ii) apply new network access credentials to connect with the owner's selected wireless network can be time consuming and difficult for users. Entering network access credentials is also prone to errors and especially difficult for devices with limited user interfaces, where a limited user interface is common for many devices supporting applications for the "Internet of Things". Loading credentials and network configuration into a device in order to obtain connectivity to a network can also be known as device provisioning or device configuration. In an attempt to address these needs to simplify or automate device provisioning, the WiFi Alliance TM released a Device Provisioning Protocol (DPP) specification on April 9, 2018 as version 1.0 (DPPv1.0), which is herein incorporated by reference in its entirety. Although the DPPv1.0 defines a series of messages between an initiator and a responder in order to support a device configuration, several unmet needs remain in the art in order to securely and easily deploy a Device Provisioning Protocol at wide scale with potentially billions of devices over many years.

To highlight one example unmet need by a DPPv1.0, WiFi capable devices currently being designed and deployed may have an expected life of more than a decade. During that timeframe, the device may need to undergo a configuration or device provisioning step more than one time. Reasons could include either (i) the device undergoes a "factory reset", or (ii) the device needs to be provisioned a second time, such as if the device owner changes or the device is moved to a different network, and other reasons exist as well. But, as stated in DPPv1.0, "threats against this bootstrapping method are possible if public keys and/or codes are reused." A need exists in the art to support a Device Provisioning Protocol in a manner where bootstrapping public keys can be securely reused. A need exists in the art for a responder bootstrap public key to be securely shared among a plurality of devices, while the responder bootstrap private key can remain secured.

In order to securely provision a device, a network may prefer to conduct mutual authentication with an initiator operating in the device that records and operates with initiator PKI keys. The device could be a high value piece of equipment, such as an automobile, health monitoring equipment in a hospital, or industrial equipment, etc. that could be targets for potential hackers seeking to "provision" the devices to a network under their control or a network providing the hackers access to the device. In these cases, an owner or user of a device may prefer that a device provisioning protocol use mutual authentication. The DPPv1.0 supports mutual authentication in order to securely authenticate a device and an initiator before transferring network access credentials to the device, but creates a new and

shifted problem of both (i) securely providing a responder bootstrap public key to an initiator and (ii) keeping the responder bootstrap private key secure. In other words, recording a responder bootstrap public key for a responder that is located nearby the initiator can be difficult if there is no pre-existing relationship or communication between a device with an initiator and a responder before the start of a device provisioning protocol. In addition, an insecure responder may have no pre-existing or sufficiently secured relationship with a network, yet the network may need to securely transfer credentials for the network to the device through the insecure responder. A need exists in the art to securely and readily enable an initiator to have access to an responder bootstrap public key, while also enabling a responder to have access to the corresponding initiator bootstrap public key. A need exists in the art for a network to securely transfer credentials to a device through an insecure responder.

With the widespread deployment of WiFi access points, a device may have access to potentially a plurality of different access points associated with potentially different networks. Although DPPv1.0 supports configuration of a device, DPPv1.0 does not suggest how a device could be pre-configured to support each of the potential plurality of different networks in order to support a device provisioning protocol. Also, DPPv1.0 does not suggest how a pre-configured device could attempt to communicate with the potential plurality of different access points in order to obtain at least one set of network credentials from the potential plurality of different networks. In other words, a need exists in the art for an initiator to use a pre-configuration in order to automatically and securely receive a set of credentials for the device without having to separately conduct an “out of band” receipt of a responder bootstrap public key when the device is taken to the vicinity of a supported network. A need exists in the art for (i) the credentials to be encrypted in a manner such that the responder cannot feasibly read the credentials, where (ii) the device can receive the encrypted credentials and convert the encrypted credentials into plaintext form in order for the device to load and use the plaintext credentials with a network.

A network can use DPPv1.0 in order to connect devices to the network using a responder. The security for conducting a device provisioning protocol according to DPPv1.0 can depend on the security of a responder, since as envisioned by DPPv1.0 the responder (i) records a responder bootstrap private key and (ii) conducts key exchanges with the responder bootstrap private key. However, a responder for conducting a DPP may be outside the control of a network and may also be an insecure device. A responder could be a “rooted” mobile phone operated by a hacker or also a “rouge” access point, as a couple of examples, and many other examples exist as well. Or, the responder could simply be operated by a computing device that is insecure or operates with security policies that do not meet the standards of the network. A need exists in the art for an insecure responder to conduct a device provisioning protocol that is compatible with DPPv1.0, while simultaneously keeping (i) the responder bootstrap private key and (ii) cryptographic operations with the responder bootstrap private key under the control or within the security policies of a network providing credentials to the device.

Many other examples exist as well for needs in the art to support efficient yet secure device configuration or provisioning, and the above are examples are just a few and intended to be illustrative instead of limiting.

SUMMARY

Methods and systems are provided for a networked responder proxy to conduct a device provisioning protocol (DPP) with an initiator operating in a device. The device provisioning protocol can support the Device Provisioning Protocol specification version 1.0 and subsequent versions from the WiFi Alliance™. Conducting a DPP with a device can transfer a set of network credentials to the device, in order for the device to connect with a network using the network credentials. The set of network credentials can also include a network configuration, where the network configuration includes settings to use with a WiFi radio in the device, in order for the WiFi radio to communicate with other nodes with the credentials, including an access point which can provide connectivity to an IP network for the device.

A device can operate an initiator, which periodically automatically transmits DPP authentication request messages until receiving a DPP authentication response from a responder proxy. The initiator can record and operate an initiator configuration, specifying a frequency band to utilize, a wireless networking technology to implement (e.g. 802.11 ac, 802.11ah, 4G LTE, 5G, etc.) and a channel list. The wireless networking technology for an initiator and a responder proxy can support a wide-area network (WAN) as well, such as 4G and 5G networks associated with public land mobile networks. The initiator can record an initiator bootstrap public key, an initiator bootstrap private key, and, when supporting mutual authentication with a responder, a responder bootstrap public key. The PKI keys can be associated with a selected set of cryptographic parameters, where the parameters specify settings to use for cryptographic operations by the initiator. The initiator can also include a random number generator, a PKI key pair generation algorithm, a secure hash algorithm, an elliptic curve Diffie Hellman key exchange algorithm, a symmetric ciphering algorithm, a digital signature algorithm, and an asymmetric ciphering algorithm.

A network can operate an access point, where the access point operates with a set of credentials. The credentials can include an identifier for the network such as an SSID, a pairwise master key (PMK) or a pre-shared key (PSK), a pairwise master key identifier (PMKID), and also a configuration. Credentials for other authentication schemes between a device and an access point could be supported as well, including EAP-TLS, EAP-PSK, EAP-AKA, EAP-AKA', etc. The access point can support wireless networking standards in the series of 802.11 standards. A network can also operate a plurality of access points. The network can operate or be associated with a Device Provisioning Protocol (DPP) server, and a device database. The network can be connected to an IP network, which can comprise the globally routable public Internet. Different servers within a network can be connected using an Intranet, a virtual private network (VPN), Internet Protocol Security (IPSEC), or similar techniques to secure communication between the servers. The access point for a

network can also comprise a enodeb for 4G wireless technology or a gnodeb for 5G wireless networking technology. The credentials with 4G networking technology can include at least a key K and the credentials with 5G networking technology can include at least a key Kausf.

The network can operate or communicate with the device database. The DPP server in the network can also communicate with the device database. The device database can record information for a plurality of devices before initiators in the device conduct the device provisioning protocol with a responder proxy. The device database can record the device identity, an initiator identity, which could be in the form of an initiator media access control (MAC) address, and a key table. The key table in a device database can record for each device an initiator bootstrap public key, a responder bootstrap public key, and a responder bootstrap private key. Recorded data within a device database can be obtained by a network (i) upon manufacturing from a device manufacturer, or (ii) before distribution of a device to the location of the network's access point.

The DPP server for a network can operate a server database, where the server database records information for a plurality of devices connecting with the network. The server database can record the device identity, media access control (MAC) addresses for devices and initiators, a responder mode, a selected set of cryptographic parameters for each device, a responder bootstrap private key, the initiator bootstrap public key, the responder bootstrap public key, a responder ephemeral public key, and responder ephemeral private key, a public key exchange (PKEX) key, a first, second, and third derived shared secret keys for an initiator. Depending on the responder mode, not all PKI keys and secret shared keys from the previous sentence may be recorded in the server database. The DPP server can also include a set cryptographic algorithms, where the cryptographic algorithms can include a random number generator, a PKI key pair generation algorithm, a secure hash algorithm, an elliptic curve Diffie Hellman key exchange algorithm, a symmetric ciphering algorithm, a digital signature algorithm, and an asymmetric ciphering algorithm. The cryptographic algorithms within a DPP server can be compatible with the equivalent set of cryptographic algorithms in the responder and also the initiator.

A responder can be a computing device that includes a WiFi radio and can be associated with a network. The responder can operate the WiFi radio with a responder configuration in order for the responder to send and receive messages with the initiator. The responder can be located in sufficient proximity with the initiator so that the WiFi radio for the initiator and the WiFi radio for the responder can communicate using 802.11 standards. The responder can operate a DPP application, where the DPP application can also include the set of cryptographic algorithms, which are compatible with cryptographic algorithms in both the initiator and the DPP server. The responder can use an IP network in order to communicate with the network via the IP network.

In a first responder mode, the responder can establish a secure, mutually authenticated connection with a DPP server operating in a network. The DPP server can send configuration data to

the responder in preparation for the responder operating as a responder proxy to receive incoming DPP authentication requests from at least one initiator. The responder can conduct a responder configuration step after receiving the responder configuration data from the DPP server. An initiator can receive and/or record a responder bootstrap public key and conduct a series of steps to transmit a DPP authentication request, potentially as a broadcast message on a WiFi channel on which the device operating the responder listens. The initiator can record an initiator key table and an initiator DPP configuration to create and transmit a series of DPP authentication request messages on different channels within the initiator DPP configuration. The responder can receive at least one DPP authentication request message transmitted by the initiator using a channel in a responder DPP configuration. The responder can forward data from the DPP authentication request received from the initiator to the DPP server through the secure session.

Continuing in the first responder mode, the DPP server can then conduct a series of steps in order to process data in the Device Provisioning Protocol (DPP) authorization request received by the responder operating as a responder proxy. The DPP server can derive a responder ephemeral PKI key pair. The DPP server can query a device database for data for the device using either the initiator media access control (MAC) address or the hash value of the initiator bootstrap public key from the DPP authentication request message. The DPP server can conduct a first responder elliptic curve Diffie-Hellman (ECDH) key exchange with the initiator bootstrap public key and the derived responder private key to derive a key k1. The DPP server can use the derived key k1 to decrypt ciphertext from the DPP authentication message and read a plaintext initiator authentication value.

Continuing in the first responder mode, the DPP server can conduct a second responder ECDH key exchange with the received initiator ephemeral public key and the derived responder private key to derive a key k2. The DPP server can conduct a third responder ECDH key exchange with the initiator bootstrap public key and a combination of the derived responder private key and the responder bootstrap private key in order to derive a key ke. The DPP server can derive a responder nonce and calculate a responder authentication value using the received and decrypted initiator authentication value. The DPP server can encrypt the responder authentication value into a first responder ciphertext using the derived key ke. The DPP server can (i) encrypt the first responder ciphertext and the responder nonce and the initiator nonce using the derived key k2 into (ii) a second responder ciphertext. The DPP server can send the responder a message with the second responder ciphertext and the derived responder ephemeral public key. The responder operating as a responder proxy can send the initiator a DPP authentication response message, which includes the second responder ciphertext and the responder ephemeral public key. Note that the responder operating as a responder proxy does not need to record the responder bootstrap private key.

Continuing in the first responder mode, the initiator can receive the DPP authentication response from the responder operating as a responder proxy and perform a series of steps to decrypt

the received second responder ciphertext. The initiator can conduct an initiator ECDH key exchange using the responder ephemeral public key and the initiator ephemeral private key in order to derive the key k_2 . The initiator can conduct a second initiator key exchange using (i) a combination of the responder bootstrap public key and the responder ephemeral public key and (ii) the initiator bootstrap private key and in order to derive the key k_e . The initiator can decrypt the second responder ciphertext using the derived key k_2 . The initiator can read the plaintext responder nonce and the first responder ciphertext. The initiator can decrypt the first responder ciphertext using the derived key k_e . The initiator can read the plaintext responder authentication value from the first responder ciphertext. The initiator can calculate the responder authentication value, and if the received responder authentication value matches the calculated responder authentication value, then the responder is authenticated.

The initiator can calculate an initiator authentication value using at least the received and decrypted responder nonce. The initiator can encrypt the initiator authentication value using the key k_e , thereby creating an initiator ciphertext. The initiator can send a DPP authentication confirm message to the responder with the initiator ciphertext. The responder operating as a responder proxy can forward the initiator ciphertext to the DPP server through the secure session. The DPP server can decrypt the initiator ciphertext using the derived key k_e and read the plaintext initiator authentication value. The DPP server can calculate the same initiator authentication value, and if the received, decrypted initiator authentication value matches the calculated authentication value then the initiator is authenticated with the DPP server and the responder proxy.

After the mutual authentication, the responder can begin operating as a configurator and the initiator can begin operating as an enrollee. The initiator can generate an enrollee nonce and also a configuration attribute, where the configuration attribute describes WiFi capabilities for the device. The initiator can create a second initiator ciphertext by encrypting the enrollee nonce and the configuration attribute with the derived key k_e . The initiator can send a DPP configuration request to the responder, where the DPP configuration request can include the second initiator ciphertext. The responder operating as a responder proxy can send the second initiator ciphertext to the DPP server.

The DPP server can receive the second initiator ciphertext from the responder operating as a responder proxy and decrypt the second initiator ciphertext using the derived key k_e , in order to read the plaintext enrollee nonce and the configuration attribute. Using the configuration attribute, the DPP server can select a set of network credentials for the device operating the initiator. The DPP server can encrypt the received enrollee nonce and the network credentials using the derived key k_e , creating a third responder ciphertext. The DPP server can send the third responder ciphertext to the responder operating as a responder proxy. The responder can forward the third responder ciphertext to the initiator in a DPP configuration response message.

The initiator can receive the DPP configuration response message and decrypt the third responder ciphertext using the derived key k_e in order to read the plaintext network credentials. The

initiator can pass the network credentials to the device, which can apply the credentials with the WiFi radio for the device. The device can subsequently connect with a network access point using the received network credentials. The network access point can provide connectivity to an IP network for the device. The DPP server and the device can record that the DPP session is successfully completed.

5 These as well as other aspects and advantages will become apparent to those of ordinary skill in the art by reading the following detailed description, with reference where appropriate to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Various exemplary embodiments are described herein with reference to the following drawings, wherein like numerals denote like entities.

5 Figure 1a is a graphical illustration of an exemplary system, where a network operates a responder and communicates with an initiator in a device, in order to conduct a device provisioning protocol, in accordance with exemplary embodiments;

Figure 1b is a graphical illustration of a device provisioning protocol for authentication of a responder and configuration of an initiator, in accordance with conventional technology;

10 Figure 1c is a graphical illustration of a device provisioning protocol for mutual authentication of an initiator and a responder, and configuration of an initiator, in accordance with conventional technology;

Figure 1d is an illustration of an exemplary initiator DPP configuration and initiator key database, with tables recording exemplary data for a plurality of bootstrap responder public keys and bootstrap initiator private keys, in accordance with exemplary embodiments;

Figure 1e is an illustration of an exemplary responder bootstrap public keys and initiator bootstrap PKI keys recorded by a plurality of devices, in accordance with exemplary embodiments;

Figure 1f is a graphical illustration of hardware, firmware, and software components for a responder, including a responder configuration step, in accordance with exemplary embodiments;

20 Figure 2a is an illustration of an exemplary device database, with tables for a device database recording exemplary data, in accordance with exemplary embodiments;

Figure 2b is a flow chart illustrating exemplary steps for an initiator to use an initiator DPP configuration and initiator key database in order to automatically connect with a responder proxy, in accordance with exemplary embodiments;

25 Figure 3a is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device database, a DPP server, a responder proxy, and an initiator, in accordance with exemplary embodiments;

Figure 3b is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a DPP server, a responder proxy, and an initiator, in accordance with exemplary embodiments;

30 Figure 3c is an illustration of an exemplary server database, with tables for a server database recording exemplary data, in accordance with exemplary embodiments;

Figure 3d is a flow chart illustrating exemplary steps for creating and verifying a digital signature using PKI keys, parameters, and data input, in accordance with exemplary embodiments;

Figure 3e is a flow chart illustrating exemplary steps for using asymmetric ciphering in order to encrypt credentials using a public key and decrypting credentials using the corresponding private key, in accordance with exemplary embodiments;

Figure 4a is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key, and for using the derived shared secret key to encrypt and decrypt data, in accordance with exemplary embodiments;

Figure 4b is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key, and for using the derived shared secret key to encrypt and decrypt data, in accordance with exemplary embodiments;

Figure 4c is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key, and for using the derived shared secret key to encrypt and decrypt data, in accordance with exemplary embodiments;

Figure 4d is a flow chart illustrating exemplary steps for conducting key exchanges using PKI keys in order to derive a first and a second shared secret keys, where ECC point addition is performed on output from a key exchange algorithm in order to derive a third shared secret key;

Figure 5a is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device database, a DPP server, a responder proxy, and an initiator, in accordance with exemplary embodiments;

Figure 5b is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device database, a DPP server, a responder proxy, and an initiator, in accordance with exemplary embodiments;

Figure 6 is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a DPP server, a responder proxy, and an initiator, in accordance with exemplary embodiments;

Figure 7 is a graphical illustration of an exemplary system, where an access point with an initiator communicates with a responder proxy using a hosted device provisioning protocol, in accordance with exemplary embodiments; and,

Figure 8 is a graphical illustration of an exemplary system, where an access point operates as a responder proxy for a device with an initiator, in order transfer a set of network credentials to the device, in accordance with exemplary embodiments.

DETAILED DESCRIPTION**Figure 1a**

Figure 1a is a graphical illustration of an exemplary system, where a network operates a responder and communicates with an initiator in a device, in order to conduct a device provisioning protocol, in accordance with exemplary embodiments. The system 100 can include a network 107 and a device 106. Network 107 can operate a WiFi access point 105, a DPP server 103, and a device database 104. The nodes within network 107 can communicate via an IP network 113. Device 106 can be a computing device operating an initiator 101 and a WiFi radio 106r. Device 106 can include a device identity 106i for a network 107 to track or identify a device 106 across a plurality of different devices 106. Or, in exemplary embodiments a media access control (MAC) address MAC.I 106n for device 106 can be used as a device identity.

Responder proxy 102 can operate as a responder and send and receive messages with device 106 operating an initiator 101 according to a device provisioning protocol (DPP) 191 as specified in the “Device Provisioning Protocol Specification version 1.0” (DPPv1.0) from the WiFi Alliance™, which is hereby incorporated by reference in its entirety. Subsequent and related versions of a DPP 191 could be supported as well by initiator 101 in device 106 and a responder proxy 102. A summary of the message flows between responder proxy 102 and device 101 according to the DPP 191 are depicted and described in connection with Figure 1b and Figure 1c below, as well as subsequent figures herein. The message flows for a DPP 191 between responder proxy 102 and initiator 101 could be via a WiFi data link 192, where the WiFi data link 192 operated by (i) initiator 101 can use an initiator DPP configuration 130 and (ii) responder proxy 102 can use a responder DPP configuration 131. Responder proxy 102 can be associated with a tag 102v, where tag 102v can record a responder bootstrap public key Br 102a.

Tag 102v can be in the form of a QR code or NFC tag, which could be read by device 106 in order to obtain the responder bootstrap public key Br 102a “out of band”. Or, in some embodiments tag 102v can be optionally omitted, where an initiator 101 can receive or record responder bootstrap public key Br 102a via other means, such as during manufacturing or distribution of device 106 where a device 106 could record a plurality of different responder bootstrap public keys Br 102a in a responder public key table 101ta depicted and described in connection with Figure 1d below.

Responder proxy 102 could operate a WiFi radio as depicted, where the WiFi radio could record a media access control (MAC) address MAC.R 102n. In exemplary embodiments, a wireless device could operate the responder proxy 102, where the wireless device could comprise a mobile handset, a smartphone, a wireless router, a tablet computer, and/or a wireless gateway. Other

possibilities exist as well for the embodiments of a wireless device operating a responder proxy 102 without departing from the scope of the present disclosure. MAC.R 102n for responder 102 could comprise a string of octets to identify responder 102 with any of a WiFi access point 105, device 106, network 107, and DPP server 103. In order to acquire the data to function as a responder according to a DPP 191 and DPPv1.0, responder 102 in Figure 1a can communicate and transmit/receive data from network 107 via IP network 113. Additional details regarding the electrical components within a responder 102 or WiFi access point 105 operating a responder 102 are depicted and described in connection with Figure 1f below. The responder proxy 102 can also include a set cryptographic algorithms, where the cryptographic algorithms can include a random number generator, a PKI key pair generation algorithm, a secure hash algorithm, an elliptic curve Diffie Hellman key exchange algorithm, a symmetric ciphering algorithm, a digital signature algorithm, and an asymmetric ciphering algorithm. The cryptographic algorithms within a responder proxy 102 can be compatible with the equivalent set of cryptographic algorithms in the initiator 101 and also the DPP server 103.

Embodiments of the present disclosure contemplate both (i) some of the functionality of a responder in the DPP specification version 1.0 operates within responder proxy 102 depicted in Figure 1a, and (ii) other functionality of a responder according to the DPP protocol can be performed or conducted by DPP server 103. In other words, (x) from the perspective of device 106, responder proxy 102 can operate as a responder according to the DPP Specification, but (y) in the present disclosure some of the functionality and data for a responder as contemplated by the DPP Specification remains in the DPP server 103. Depictions and descriptions below in the present disclosure may utilize the term “responder 102” or configured “responder 102’ ”, which can also comprise a “responder proxy 102” depicted in Figure 1a. As contemplated herein, a “responder proxy 102’ ” can be equivalent to “responder 102’.”

This configuration of operating a responder proxy 102 in a system 100 (along with proper operation of a network 107) can address and solve the needs in the art described above. A responder proxy 102 can communicate with DPP server 103 via IP network 113. Although a responder proxy 102 is depicted as operating in WiFi access point 105 in Figure 1a, other embodiments of the present disclosure contemplate that different WiFi capable devices than access point 105 could operate a responder proxy 102, such as a mobile phone 701 depicted in Figure 7 below. Other possibilities exist as well for a WiFi capable device to operate a responder proxy 102 without departing from the scope of the present disclosure. In addition, although the use of a WiFi radio is depicted in Figure 1a and other figures herein, other wireless networking technologies and radios besides WiFi could be used with responder proxy 102 and device 106 with initiator 101. As one example, (i) a responder proxy could operate with a wireless wide area network (WAN) radio such as a “gnodeb” with a 5G network and (ii) device 106 with initiator 101 could operate a radio module supporting 5G networking standards.

IP network 113 could be either a Local Area Network (LAN) or a Wide Area Network (WAN), or potentially a combination of both. IP network 113 could comprise a packet switched network supporting either IEEE 802.11 (WiFi) standards and/or IETF standards for IP networking such as IPv4 or IPv6. Responder 102 within access point 105 and network 107 could also utilize a variety of WAN
 5 wireless technologies to communicate, including Low Power Wide Area (LPWA) technology, 3rd Generation Partnership Project (3GPP) technology such as, but not limited to, 3G, 4G Long-Term Evolution (LTE), or 4G LTE Advanced, NarrowBand – Internet of Things (NB-IoT), LTE Cat M, proposed 5G networks, and other examples exist as well. A wired responder proxy 102 can connect to the IP network 113 via a wired connection such as, but not limited to, an Ethernet, a fiber optic, or
 10 a Universal Serial Bus (USB) connection (not shown). In some exemplary embodiments, access to an IP network 113 for responder proxy 102 can be provided by access point 105, where access point 105 may also record and operate with a set of network credentials 109.

IP network 113 could also be a public or private network supporting Internet Engineering Task Force (IETF) standards such as, but not limited to, such as, RFC 786 (User Datagram Protocol), RFC
 15 793 (Transmission Control Protocol), and related protocols including IPv6 or IPv4. A public IP network 113 could utilize globally routable IP addresses, and a private IP network 113 could utilize private IP addresses which could also be referred to as an Intranet. Other possibilities for IP Network 113 exist as well without departing from the scope of the disclosure. Although not depicted in Figure 1a, elements within a system 100 could each operate or be associated with a firewall, such that packets
 20 from IP network 113 are filtered or restricted in order to secure the communication between the nodes.

Device 106 can comprise a computing device with a WiFi radio for transmitting and receiving data such as via WiFi data link 192. Device 106 can include a WiFi radio 106r. Radio 106r and radio 102i for responder proxy 102 can also support other wireless technologies besides WiFi, including 4G and 5G networks. WiFi radio 106r can operate with a media access control (MAC) address MAC.I
 25 106n. Responder 102 can read the value MAC.I 106n from messages transmitted by device 106 over WiFi link 192. Device 106 can operate an initiator 101 and, after a configuration step 106a, also operate a set of network WiFi credentials 109 that are compatible with an access point 105. Device 106 could be powered via any of (i) traditional “wall power” potentially with an AD/DC adapter, (ii) a battery which may be periodically recharged or replaced, (iii) power over a wired LAN connection
 30 such as “power over Ethernet”, and other possibilities exist as well. A device 106 can record a certificate cert.device 106m, which can record a device public key, which can be associated with a device private key SK.device 106s also recorded in device 106. For the system depicted in Figure 1a, device 106 operating an initiator 101 can have a roll of an enrollee, and responder proxy 102 can operate with a roll of a configurator.

35 The initiator 101 in device 106 can record an initiator key database 101t, which could comprise a database such as SQLite, or alternatively simply a text file of keys and identities, and other

possibilities exist as well for the form and structure of keys recorded in an initiator 101. In exemplary embodiments, the initiator key database 101t can be stored in a secure processing environment or a secure enclave for a CPU operating in device 106. The initiator key database 101t can optionally include an initiator bootstrap public key Bi 101a, an initiator bootstrap private key bi 101b (for 5 embodiments supporting a mutual authentication 142 depicted in Figure 1c below), a responder bootstrap public key Br 102a, a set of cryptographic parameters 199a for the PKI keys, and optionally a public key exchange protocol (PKEX) shared secret key 198. The initiator 101 in device 106 can comprise a process or running software program operated by a processor and recorded in memory for device 106. Although not depicted in Figure 1a, an initiator 101 can also include a random number 10 generator, a PKI key pair generation algorithm, a secure hash algorithm, an elliptic curve Diffie Hellman key exchange algorithm, and a symmetric ciphering algorithm, a digital signature algorithm, and an asymmetric ciphering algorithm. The cryptographic algorithms within an initiator 101 can be compatible with the equivalent set of cryptographic algorithms in a responder proxy 102 and also DPP server 103.

15 A network 107 can operate an access point 105, where the access point operates with a set of credentials 109. The credentials 109 can include an identifier for the network such as an SSID.network-AP 109a, at least one pairwise master key (PMK) PMK.network-AP 109b or a pre-shared key (PSK), a pairwise master key identifier (PMKID) (not shown), and also a configuration Config.network-AP 109c. The access point can support wireless networking standards in the series of 802.11 standards. 20 Credentials 109 for other authentication schemes between a device and an access point could be supported as well, including EAP-TLS, EAP-PSK, EAP-AKA, EAP-AKA', etc. A network 107 can also operate a plurality of access points 105. In addition, access point 105 can comprise a plurality of different WiFi radios to collectively function as one access point 105, such as with "mesh" WiFi networks. Or, access point 105 can operate as a single WiFi router. The network 107 can include or 25 be associated with a Device Provisioning Protocol (DPP) server 103, and a device database 104. Different servers within the network 107 can be connected using an Intranet, a virtual private network (VPN), Internet Protocol Security (IPSEC), or similar techniques to secure communication between the servers.

The network 107 can operate or communicate with the device database 104. The DPP server 30 103 in the network 107 can also communicate with the device database 104. The device database 104 can record information for a plurality of devices 106 before an initiator 101 for the device 106 conducts the device provisioning protocol 191 with the responder proxy 102. The device database 104 can record an identity for device 106, which could be in the form of either (i) an initiator media access control (MAC) address MAC.I 106n (depicted) or a hash value 101aa for the initiator bootstrap public key Bi 101a (where the hash value 101aa is not shown in Figure 1a but is shown in Figure 1b and 35 other Figures below).

A device database 104 can record at least one responder bootstrap public key 102a with a corresponding responder bootstrap private key 102b, and optionally an initiator bootstrap public key 101a for embodiments supporting a mutual authentication depicted in Figure 1c and other Figures below. Recorded data within a device database 104 can be obtained by a network 107 from either (i) upon manufacturing from a device manufacturer, or (ii) before distribution of a device to the location of the network's access point. Some data in a device database 104 could also be acquired by a responder 102 or a network 107 prior to the start of a DPP 191. For example, the derivation and recording of responder bootstrap private key br 102b can be independent of recording data in a database 104 for a device 106 (such as MAC.I 106n, key Bi 101a, etc.). Device database 104 can also record a set of cryptographic parameters 199a associated with the PKI keys for responder 102 and initiator 101. Additional exemplary data for a device database 104 is depicted and described in connection with Figure 2a below.

The DPP server 103 for a network 107 can operate a server database 103a, where the server database 103a records information for a plurality of devices 106 and a plurality of responders proxies 102 connecting with the network 107. A server database 103a can also record data for initiators 101 in devices 106, as depicted and described in connection with Figure 3c below. Depending on a responder mode 306a in Figures 2a and 3c below, only selected sets of PKI keys and secret shared keys for initiators 101 may be recorded in the server database 103a. The DPP server 103 can also include a set cryptographic algorithms, where the cryptographic algorithms can include a random number generator, a PKI key pair generation algorithm, a secure hash algorithm, an elliptic curve Diffie Hellman key exchange algorithm, a symmetric ciphering algorithm, a digital signature algorithm, and an asymmetric ciphering algorithm. The cryptographic algorithms within a DPP server 103 can be compatible with the equivalent set of cryptographic algorithms in the initiator 101 and also the responder proxy 102.

Note that the security of communications between a device 106 and an access point 105 using credentials 109 may generally depend on the security of a configuration step 106a, which in turn may rely on a device provisioning protocol 191. Further, a manufactured device 106 may not have the credentials 109 or configuration in order to communicate with an access point 105 without successfully completing a DPP 191 and a configuration step 106a (or other manual or cumbersome means) in order for device 106 to securely receive and use network credentials 109. After a DPP 191 and a configuration step 106a described in the present disclosure and in multiple Figures below, device 106 can communicate with access point 105 using credentials 109.

Network 107 can also be associated with an external network 108, which can comprise a network similar to network 107. External network 108 could operate a plurality of access points 105, and in exemplary embodiments although the device provisioning protocol 191 is conducted with network 107, the credentials 109 transferred to device 106 in a configuration step 106a may belong to

the external network 108, and the device 106 could connect with an access point operated by the external network 108 using the credentials 109 instead of connecting with access point 105 operated by network 107. For these embodiments, network 107 could receive the credentials 109 for device 106 as asymmetrically encrypted credentials 109' (described in Figure 3b below) from the external network 108.

Figure 1b

Figure 1b is a graphical illustration of a device provisioning protocol for authentication of a responder and configuration of an initiator, in accordance with conventional technology. Figure 1b depicts a summary of the WiFi Device Provisioning Protocol (DPP) specification, version 1.0 which was published on April 9, 2018. The summary depicted in Figure 1b highlights recorded bootstrap PKI keys, derived ephemeral PKI keys, and messages transmitted and received between an initiator 101 and a responder 102*. The use of keys and messages for a DPP 191 can accomplish both (i) authenticating the responder 102* and (ii) transfer a configuration object to initiator 101 where the configuration object could comprise a set of network access credentials 109 from Figure 1a. Responder 102* is depicted with a "*" and has several differences than a responder proxy 102 depicted in Figure 1a and subsequent Figures below, although some functionality is shared between a responder 102* in Figure 1b and responder proxy 102 herein. Both responder 102* and responder proxy 102 (from Figure 1a and other Figures below) can send and receive the full and complete set of a DPP messages for a Device Provisioning Protocol 191. As contemplated herein, a responder 102* can comprise a responder operating according to the conventional technology depicted in Figure 1b and Figure 1c, while a responder 102 or responder 102' or responder proxy 102' can (i) operate as a proxy for DPP 191, and (ii) record and operate on different PKI keys than those depicted for responder 102* in Figure 1b.

In other words, responder 102* and a responder proxy 102 can send equivalent messages such that a initiator 101 (i) would process the messages in the same manner and (ii) would not normally be able to detect to discern whether a responder proxy 102 or an responder 102* was transmitting and receiving messages for the Device Provisioning Protocol 191. In the present disclosure, responder proxy 102 (not depicted in Figure 1b, but depicted in Figure 1a, Figure 1f, Figure 3a, etc.) can send and receive DPP messages 122, 123, etc. with initiator 101 as an responder according to the DPP Specification 1.0. Responder 102* or a responder proxy 102 can be a process, program, firmware, or software application operating in a computing device with a WiFi compatible radio, in order to communicate with an initiator 101 in a device 106 via a WiFi

data link 192 (in Figure 1a). Exemplary additional configurations for an initiator 101 and a responder proxy 102 within computing devices is depicted and described in connection with Figures 7 and 8 below, and other possibilities exist as well without departing from the scope of the present disclosure.

5 Other differences between a responder 102* and a responder proxy 102 can exist in the present disclosure. For a system 100 with a responder proxy 102 in Figure 1a and Figures below, the recordation of PKI keys, PKI key derivation, and cryptographic processes such as symmetric encryption and decryption can be distributed between a responder proxy 102 and a DPP server 103. However, a responder 102* can internally record the keys necessary for symmetric key
10 exchange, key derivation, and symmetric encryption and decryption, as contemplated by DPP specification version 1.0. The distributed storage and operation with PKI keys for a system that includes a responder proxy 102 is depicted and described in more detail in connection with Figure 3a, Figure 3b, Figure 3c, Figure 5a, Figure 5b, and Figure 6 below. Additional details for internal components, memory structures, values, and operation of a responder proxy 102 in order to
15 provide equivalent functionality as responder 102* for an initiator 101 are described in Figures below.

Initiator 101 can record ephemeral keys and derive symmetric encryption keys in order to authenticate the responder 102* and receive the network access credentials 109 according to a Device Provisioning Protocol 191. For a system with responder authentication 141, an initiator
20 101 can record a media access control (MAC) address MAC.I 106n. MAC.I 106n can (i) be associated with a WiFi radio operating in device 106 that also operates initiator 101 and (ii) comprise a string of octets in order to uniquely identify initiator 101 and/or device 106 within a wireless network. MAC.I 106n can also comprise an identity for device 106 with network 107 in Figure 1a. With current, convention technology, MAC.I 106n can comprise a string of 6 octets,
25 although other possibilities exist as well for a MAC.I 106n. Initiator 101 can also operate a key pair generation function 101y in order to derive ephemeral PKI keys such as an initiator ephemeral public key P_i 101c and an initiator ephemeral private key p_i 101d. As contemplated herein, an ephemeral PKI key can also be referred to as a “protocol” key. In other words, key P_i 101c can be referred to as either (i) an initiator protocol public key P_i 101c or (ii) an initiator ephemeral
30 public key P_i 101c, which can both represent the same numeric value, number, or sequence of digits.

A key pair generation function 101y or 102y could use (i) a random number generator in order to derive the PKI private key and (ii) a selected set of cryptographic parameters 199a (depicted in Figure 1a above and Figure 2a below and other Figures herein) in order to specify an

elliptic curve cryptography (ECC) curve name, key length, key formatting, etc. As contemplated herein, the algorithms used with ECC PKI keys in the present disclosure can be compatible and substantially conform with ECC algorithms and keys as specified in (i) the IETF Request for Comments (RFC) 6090 titled “Fundamental Elliptic Curve Cryptography Algorithms”, and (ii) IETF RFC 5915 titled “Elliptic Curve Private Key Structure”, and also subsequent and related versions of these standards.

With a random number generated by responder 102* or a responder proxy 102, a key pair generation function 102y, a derived ephemeral responder private key pr 102r can be used to derive a corresponding ephemeral initiator public key Pr 102p. The two derived PKI keys can be used in conjunction for subsequent operations such as an Elliptic Curve Diffie Hellman key exchange and also other cryptographic algorithms such as a key derivation function and a digital signature algorithm. As contemplated in the present disclosure, the use of a capital letter as the first character for a PKI key can represent a public key, the use of a lower case letter as the first character for a PKI key can represent a private key. As contemplated in the present disclosure, the second letter for a PKI key can represent the entity the key is associated with or belongs to (e.g. “r” for responder 102 and “i” for initiator 101). Further, the letter “B” or “b” as the first character for a PKI key can represent a recorded bootstrap key, which could be static or relatively static over the lifetime of a responder or initiator, and the letter “P” or “p” the first character for a PKI key can represent an ephemeral key, which can also be referred to as a protocol key.

Responder 102* in Figure 1b and Figure 1c can include functionality to operate as a responder according to the DPP Specification version 1.0 for the purpose of transmitting and receiving messages according to the DPP protocol 191. For a DPP 191, a responder 102* can record a responder bootstrap public key Br 102a, a responder bootstrap private key 102b, a MAC address MAC.R 102n, and also operate a key pair generation algorithm 102y. A responder bootstrap public key Br 102a and responder bootstrap private key br 102b recorded in responder 102* could comprise an ECC key pair selected and formatted to be compatible with the set of parameters 199a for initiator 101 (in Figure 1a above). Ephemeral bootstrap PKI keys for responder 102* could also be formatted and selected with a compatible set of cryptographic parameters 199a for Pi 101c and pi 101d. Bootstrap PKI keys for responder 102* could be recorded in responder 102* or nonvolatile memory for a WiFi enabled computing device operating a responder 102*. With the conventional technology depicted in Figure 1b, the bootstrap PKI keys could be recorded in responder 102* during manufacturing or a configuration of responder 102* before receiving the DPP authentication request message 122. MAC address MAC.R 102n

can comprise an identity of responder 102* or the computing device operating responder 102* on a WiFi network, and can be similar to MAC.I 106n for an initiator 101 as described above.

An initiator 101 and a responder 102* can conduct a Device Provisioning Protocol (DPP) 191. Again, as contemplated herein, an exemplary Device Provisioning Protocol 191 according to conventional technology could comprise the “Device Provisioning Protocol Specification” version 1.0 as published by the WiFi Alliance™. In message 121, an initiator 101 can receive the responder bootstrap public key Br 102a for responder 102*. The “out of band” transfer of Br 102a in message 121 could comprise several different possible methods for initiator 101 to receive Br 102a, which are also described in the DPP specification 1.0, such as via reading a QR code on the device operating responder 102*, reading a near-field communications tag, using a Bluetooth wireless connection to read Br 102a from responder 102*, and other possibilities exist as well. The data could be read via a tag 102v for responder 102 depicted and described in connection with Figure 1a. A DPP 191 also contemplates that associated information for responder 102* could be transferred in message 121, such as an identity for the WiFi enabled computing device operating responder 102*, MAC.R 102n, in addition to Br 102a and parameters 199a in message 121, as well as a responder DPP configuration 131 as depicted in Figure 1a.

Initiator 101 can receive Br 102a from message 121 and derive PKI keys Pi 101c and pi 101d. Initiator 101 can conduct an Elliptic Curve Diffie Hellman key exchange algorithm 401a as depicted and described below in key exchange 314 step for Figure 4a in order to derive a first shared secret key with responder 102*. Initiator 101 can derive a random number comprising an initiator nonce and also determine a set of capabilities for the initiator, such as a configurator or enrollee as specified in DPP specification version 1.0. Initiator 101 can encrypt the initiator nonce and the initiator capabilities using the derived shared secret key as depicted and described in connection with an encryption step 315 in Figure 4a below. Initiator 101 can send the derived public key Pi 101c and a ciphertext with the encrypted data in a message 122. Although not depicted in Figure 1b, initiator 101 can also send a hashed value of responder bootstrap public key Br 102a in message 122.

As specified in DPP specification version 1.0, message 122 could also include a hashed value for initiator bootstrap public key Bi 101a. But, for a responder authentication 141 depicted in Figure 1b, the hashed value of Bi 101a may optionally not be calculated or sent in message 122 (since responder 102* may not record the corresponding initiator bootstrap public key Bi 101a for a responder authentication 141 as depicted in Figure 1b). The hash algorithm on a bootstrap public key could comprise the SHA-256 hash algorithm according to IETF RFC 5754 titled “Using SHA2 Algorithms with Cryptographic Message Syntax”. Other hash algorithms could be used as

well and specified in a selected set of cryptographic parameters 199 as depicted and described in connection with Figure 2b below, such as SHA-3, SHA-384, etc. Message 122 can comprise a Device Provisioning Protocol Authentication Request according to DPP specification version 1.0. Additional details for a message 122 are provided in section 6.2.2 of DPP specification version 1.0.

Responder 102* can receive message 122 and follow the steps for a responder according to the section 6.2.3 of DPP specification version 1.0, in order to generate a Device Provisioning Protocol Authentication Response, which could comprise message 123. Responder 102* can conduct an Elliptic Curve Diffie Hellman key exchange algorithm 401a as depicted and described below in key exchange 319 step for Figure 4a in order to mutually derive a first shared secret key with initiator 101. Responder 102* can decrypt the initiator nonce and the initiator capabilities using the derived first shared secret key as depicted and described in connection with an decryption step 320 in Figure 4a below. Responder 102* can use the key pair generation algorithm 102y in order to derive an ephemeral responder public key Pr 102p and a corresponding ephemeral private key 102r. Responder 102* can use the derived responder ephemeral private key pr 102r and the received initiator ephemeral public key Pi 101c in order to conduct a key exchange algorithm 314b as depicted below in Figure 4b and derive a second secret shared key. The first mutually derived symmetric key can comprise k1 and the second mutually derived symmetric key can comprise k2 according to the DPP specification version 1.0. Responder 102* can derive a random number comprising a responder nonce and also determine a set of capabilities for the responder, such as a configurator as specified in DPP specification version 1.0.

Responder 102* could also generate a responder authentication value, which could comprise a hash value over at least the initiator nonce received in message 122 and the generated responder nonce. Responder 102* could use the first shared secret key and the second shared secret key in order to derive a third shared secret key. Responder 102* could encrypt the responder authentication value with the third shared secret key and also then include the encrypted responder authentication value with the responder nonce and responder capabilities in a ciphertext. The ciphertext could be encrypted with the second shared key, as depicted and described in connection with an encryption step 315b in Figure 4b below. Responder 102* could send the derived ephemeral responder public key Pr 102p and the ciphertext in a message 123.

Initiator 101 can receive message 123 and record or store the received ephemeral responder public key Pr 102p. Initiator 101 can follow the steps in section 6.2.4 of DPP specification version 1.0 in order to process message 123 and confirm the authentication response from responder 102* in message 123. Initiator 101 can mutually derive the second shared secret

key using the received ephemeral responder public key 102p and the derived initiator ephemeral private key 101d, using a key exchange algorithm 319a as depicted in Figure 4b. Initiator 101 can decrypt the ciphertext in message 123 using the second mutually derived secret shared key and a decryption step 320b in Figure 4b. Initiator 101 can derive the third secret shared key using at least the first mutually derived secret shared key and the second mutually derived secret shared key. Initiator 101 can use the third mutually derived secret shared key to decrypt the encrypted responder authentication value using a decryption step 328 in Figure 4c. Initiator 101 can calculate a value for the responder authentication, and if the calculated value for the responder authentication matches the received value for the responder authentication, then responder 102* can be considered authenticated.

Although not depicted in Figure 1b, after authenticating responder 102*, initiator 101 can send an authentication confirm message, which could comprise a DPP authentication confirm message according to section 6.2.4 of the DPP specification version 1.0. The authentication confirm message can also include an initiator authentication value, which could be encrypted with the third secret shared key. The responder 102* can receive the encrypted authentication confirm value and decrypt the value and also internally calculate the initiator authentication value. If the received initiator authentication value and the calculated authentication confirm values are equal, then the initiator is authenticated with the responder 102*. Note that for a responder authentication 141 in Figure 1b, the authentication of initiator 101 by responder 102* is weak and only comprises verifying that initiator 101 records the bootstrap public key Br 102a for responder 102*. Also, although not depicted in Figure 1b, responder 102* could have the capabilities of a configurator and receive a DPP configuration request message from initiator 101, which could take the role of an enrollee. The DPP configuration request message can be according to section 6.3.2 of the DPP specification version 1.0, and can include an enrollee nonce, which could be a random number generated by the enrollee/initiator.

As depicted in Figure 1b, responder 102* can then send a configuration object in a ciphertext in message 124. Message 124 could comprise a DPP configuration response message according to section 6.3.3 of the DPP specification version 1.0. Responder 102* can record the configuration object before sending message 124. In the present disclosure, the configuration object comprises the network credentials 109 depicted in Figure 1a above. In other words, a configuration object can be configuration information for a device 106 with initiator 101 to use in order to connect with other devices via a wireless network. Configuration object could contain a list of network identifiers, device identifiers, RF band and channel information, configuration parameters, shared keys, PKI keys, names, passwords, group temporal keys, a shared secret 198

for a PKEX key exchange, a pairwise master key (PMK) and PMK identity, etc. An exemplary list of possible values for a configuration object is depicted in table 14 of DPP specification version 1.0. In message 124, the network credentials 109 as a configuration object can be sent in a ciphertext using the third mutually derived secret shared key. As depicted in Figure 1b, the ciphertext can include both the enrollee nonce and the network credentials 109. Initiator 101 can receive message 124, decrypt the ciphertext using the third mutually derived secret shared key, and record the network credentials 109. Device 101 can then apply the network credentials 109 and communicate with access point 105 using the received network credentials 109.

Figure 1c

Figure 1c is a graphical illustration of a device provisioning protocol for mutual authentication of an initiator and a responder, and configuration of an initiator, in accordance with conventional technology. Figure 1c depicts a summary of the WiFi Device Provisioning Protocol (DPP) specification, version 1.0 which was published on April 9, 2018, supporting a mutual authentication 142 by both initiator 101 and responder 102*. The summary depicted in Figure 1c highlights recorded bootstrap PKI keys, derived ephemeral PKI keys, and messages transmitted and received between an initiator 101 and a responder 102*. Many of (i) the PKI keys for initiator 101 and responder 102*, and (ii) the messages transmitted between the nodes are equivalent to those depicted and described in connection with Figure 1b. This description of Figure 1c will focus upon the differences in order for initiator 101 and responder 102* to mutually authenticate.

In order to support a mutual authentication between initiator 101 and responder 102*, initiator 101 can record an initiator bootstrap public key Bi 101a and an initiator bootstrap private key bi 101b. Although initiator bootstrap public key Bi 101a is depicted within initiator 101, another entity besides initiator 101 could record the public key and make it available to other nodes such as a responder 102*. As contemplated by the DPP Specification, the initiator bootstrap keys can be recorded with initiator 101 before conducting a DPP 191 authentication and configuration with responder 102*. For example, the bootstrap PKI keys for the initiator could be recorded during manufacturing or distribution of the initiator, or loaded by an initiator user before conducting the depicted communications with responder 102*. Bootstrap public keys for the initiator can support a selected set of cryptographic parameters 199a, where the selected set of cryptographic parameters 199a can match the parameters used by keys in responder 102*. The DPP specification version 1.0 identifies the minimal set of cryptographic parameters in order to ensure compatibility, which can comprise the “set A” for a set of cryptographic parameters 199 depicted in Figure 2a below.

In order to conduct a mutual authentication 142 in addition to the responder authentication 141 depicted in Figure 1c, a responder 102* should have access to the initiator bootstrap public key Bi 101a. In message 126, a responder 102* can receive the initiator bootstrap public key Bi 101a for initiator 101. The “out of band” transfer of Bi 101a in message 126 could comprise several different possible methods for responder 102* to receive Bi 101a, which are also described in the DPP specification 1.0, such as via reading a QR code, reading a near-field communications tag, using a Bluetooth wireless connection to read Bi 101a from initiator 101, and other possibilities exist as well, including using WiFi direct to first establish a “peer-to-peer” WiFi connection. A DPP 191 also contemplates that associated information for initiator 101 could be transferred in message 126, such as a device identity for initiator 101, MAC.I 106n or hash value 101aa, in addition to Bi 101a. As depicted in Figure 1c, the responder 102* can also record the initiator bootstrap public key Bi 101a, which can be used to authenticate the initiator 101 as described below.

For a mutual authentication 142 between device 101 and responder 102*, the two nodes can exchange the same or equivalent messages as message 122 and message 123, which were also described above in connection with Figure 1b. For mutual authentication, the third shared encryption key ke in DPP specification version 1.0 can be derived by a responder 102* using a key exchange step 321 and the received initiator bootstrap public key Bi 101a as depicted and described in connection with Figure 4c below. The use of initiator bootstrap public key Bi 101a for the key exchange step 321 is described in section 6.2.3 of the DPP Specification version 1.0. The difference between the third mutually derived secret shared key ke for a mutual authentication 142 and a responder authentication 141 is the inclusion of initiator bootstrap public key Bi 101a in the key exchange algorithm 401c and key derivation function 402c in Figure 4c. The first mutually derived secret shared key can comprise k1 and the second mutually derived secret shared key can comprise k2. The third mutually derived secret shared key ke is used by the responder 102* to encrypt the responder authentication value as depicted in an encryption step 322 in Figure 4c. The responder authentication value can include a random number “initiator nonce” or I-nonce from the initiator 101 in message 122. The responder authentication value can comprise the value “R-auth” in step 3 of section 6.2.3 of DPP specification version 1.0.

Initiator 101 can also mutually derive the third secret shared encryption key ke using the initiator bootstrap private key bi 101b and key exchange step 327 depicted and described below in connection with Figure 4c. Initiator 101 can subsequently decrypt the responder authentication value using the mutually derived third secret shared encryption key ke and a decryption step 328 as depicted and described in connection with Figure 4c below. Initiator 101 can calculate the same value for the responder authentication value as described in section 6.2.4 of DPP specification version 1.0. If the received, decrypted responder authentication value matches the calculated responder authentication value, then the responder 102* is authenticated.

The initiator 101 can calculate an initiator authentication value using at least the received responder nonce in message 123 and encrypt the initiator authentication value using the mutually derived third secret shared key ke (from a key exchange step 327) and send the initiator authentication value in a message (not shown in Figure 1c). The responder 102* can (i) decrypt the encrypted initiator authentication value using the third mutually derived secret shared key ke (from a key exchange step 327) and (ii) also calculate the initiator authentication value. If the calculated initiator authentication value by the responder 102* matches the received, decrypted initiator authentication value then the initiator 101 is authenticated with the responder 102* for a mutual authentication. The initiator 101 as enrollee can then receive a configuration object in a message 124. Message 124 can include network credentials 109 encrypted with the third mutually derived secret shared key ke. The initiator 101 can receive message 124 and pass the decrypted network credentials 109 to device 106, which can then apply the network credentials 109 in order to communicate with network access point 105.

Figure 1d

Figure 1d is an illustration of an exemplary initiator DPP configuration and initiator key database, with tables recording exemplary data for a plurality of bootstrap responder public keys and bootstrap initiator private keys, in accordance with exemplary embodiments. Although Figure 1b and Figure 1c above depicted and described conventional technology for an initiator 101, data for an initiator 101 depicted in Figure 1d and used by initiator 101 with a responder proxy 102 is not contemplated by conventional technology. An initiator 101 communicating with a responder proxy 102 can record and operate an initiator key database 101t, where an initiator key database 101t was also depicted and described above in connection with Figure 1a. An initiator key database 101t can include a responder public key table 101ta and an initiator key table 101tb.

Responder public key table 101ta can record a plurality of bootstrap responder public keys Br 102a for a plurality of different networks 107. The present disclosure with the use of a responder proxy 102' supports the recording and operation of the corresponding a bootstrap responder private key br 102b solely within a DPP server 103 and/or a device database 104 as depicted in Figure 1a. This represents a significant difference from the conventional technology depicted in Figure 1b and Figure 1c, which contemplates that a responder 102* would operate and record bootstrap responder private key br 102b. By recording responder bootstrap private key br 102b in secure servers within network 107, the corresponding responder bootstrap public key Br 102a can be "reused" or distributed among a plurality of devices 106 during a pre-configuration step or during manufacturing or distribution, before an end user receives the device 106. The reuse of responder bootstrap public key Br 102a can comprise shared keys 102z as depicted in Figure 2a. In other words, the responder

bootstrap public key Br 102a recorded in a device 106 before the device 106 conducts a DPP 191 can comprise a pre-shared bootstrap responder public key.

In addition, a manufacturer or distributor of device 101 may not know which network 107 among a plurality of possible different networks 107 that an end user for device 106 may wish to connect device 106. Since the responder bootstrap public key Br 102a comprises a public key and can be shared, a network 107 could send the responder bootstrap public key Br 102a to the manufacturer or distributor of device 106, where the responder bootstrap public key Br 102a could be recorded in a plurality of devices 106 in nonvolatile memory. Network 107 could securely record the corresponding responder bootstrap private key br 102b in a server database 103a or a device database 104. A manufacturer or distributor of device 106 could collect the publicly shared responder bootstrap public keys Br 102a from a plurality of different networks 107 and record the resulting plurality of different responder bootstrap public keys Br 102a in a responder public key table 101ta. The responder public key table 101ta could also be recorded in a plurality of devices. In this manner, an (a) exemplary device 106 depicted in Figure 1a could record the appropriate responder bootstrap public key Br 102a for a network 107 depicted in Figure 1a before (b) the device 106 is received by an end user.

The use of a responder public key table 101ta simplifies conducting a DPP 191 between responder 102' and initiator 101, since a separate "out of band" transfer of responder bootstrap public key Br 102a in a step 121 of Figure 1b and Figure 1c could be omitted or is not required by device 106 recording key Br 102a for a responder proxy 102'. A responder proxy 102' can operate with potentially many different keys Br 102a recorded by initiator 101, since responder 102' can forward messages such as DPP 122 to DPP server 103 for processing by DPP server 103 (and DPP server 103 could record and operate on the corresponding private key br 102b).

As depicted in Figure 1d, responder public key table 101ta for an initiator 101 can include data or values for a responder bootstrap public key Br identity 101z, a responder network 107a name or identity, a responder bootstrap public key Br 102a, a secure hash value 102aa over the key Br 102a, and a set of parameters 199 for the public key Br 102a. A responder public key table 101ta could record a plurality of data for a plurality of network 107a names with corresponding key Br 102a. In exemplary embodiments, at least one responder bootstrap public key Br 102a is recorded in a table 101ta for initiator 101. In exemplary embodiments data for an initiator key database 101t can be recorded in protected nonvolatile memory within device 106 and accessible only to initiator 101 or root or privileged processes operating in a device 105.

In exemplary embodiments, an initiator 101 for a device 106 could record a plurality of responder bootstrap public keys Br 102a in a responder public key table 101ta. Reasons could include the use of different keys Br 102a for different purposes, such as a first key Br 102a for a responder authentication 141 and a second key Br 102a for a mutual authentication 142. Or, a third key Br 102a could be recorded or mapped in a tag 102v from responder 102' and received in a message 121.

Responder 101x can designate the use of select keys Br 102a with a higher level of trust, with the following exemplary results (A) and (B). For (A), device 106 could provide additional privileges for a configuration step 106a for an initiator 102' or initiator 102* that uses the second key Br 102a (e.g. a mutual authentication 142 with the second key Br 102a could require responder 102' to have key Bi 101a, and release of key Bi 101a could be under the control of device 106 or recorded only in a device database 104 in a network 107). As one example of additional privileges, a set of credentials 109 delivered to device 106 using the second key Br 102a with mutual authentication 142 using key Bi 101a could allow the installation of a new certificate authority root certificate for device 106. As one example of normal or other privileges, a set of credentials 109 delivered to device 106 using the first key Br 102a could deny the installation of a new certificate authority root certificate for device 106.

As depicted in Figure 1d, initiator key table 101tb in an initiator key database 101t can record values for an initiator bootstrap key identity 101w, a set of parameters 199 for the initiator bootstrap private key bi 101b, the initiator bootstrap public key Bi 101a, a secure hash value 101aa over the key Bi 101a, and the initiator bootstrap private key bi 101b. In exemplary embodiments, an initiator key database 101t can include at least one initiator bootstrap public key Bi 101a in order to support a mutual authentication 142 depicted in Figure 1c. One use of an initiator key table 101tb is also depicted and described in connection with Figure 2b below.

As depicted in Figure 1d, initiator key table 101tb can record data for a plurality of different initiator bootstrap PKI keys. The initiator bootstrap PKI keys could be different according to the set of parameters 199. In other words, a first key bi 101b could be for parameters 199 of "A" with a key length of 256 bits, and a second key bi 101b could be for parameters 199 of "B" with an exemplary length of 384 bits. Different elliptic curves could be specified by parameters 199 for different keys in an initiator key database 101t. In exemplary embodiments where initiator 101 can provide bootstrap public key Bi 101a "in band" to responder 102', such as, but not limited to, initiator 101 using a PKEX key exchange, then table 101tb could record a plurality of different key bi 101b and Bi 101a for the same parameters 199, and initiator 101 could select and deprecate different keys bi 101b and Bi 101a over time. One use of an initiator key table 101tb is also depicted and described on connection with Figure 2b below.

As depicted in Figure 1a, an initiator DPP configuration 130 for initiator 101 is depicted in Figure 1d. An initiator DPP configuration 130 can comprise a collection of data and values recorded in nonvolatile memory for WiFi radio 106r. Initiator DPP configuration 130 can be similar or compatible with a responder DPP configuration 131 as depicted and described in connection with Figure 1f below. Initiator DPP configuration 130 can record values for a device supported configuration 130z, a global operating class 130a and a channel list 130b. Global operating class 130a can specify a WiFi technology and frequency band for operating WiFi radio 106r, and can represent the capabilities of the radio in a manufactured device 106. Global operating class 130a can support

exemplary class values and numbers in Appendix E of the 802.11ac specification, and other possibilities exist as well without departing from the scope of the present disclosure. Channel list 130b can comprise a list of channels for initiator 101 to transmit DPP authentication request messages 122. An initiator 101 in an unconfigured device 106' can rotate through a list of Global operating classes and channel numbers in an channel list 130b and transmit a series of DPP authentication request messages 122 on each combination of global operating class 130a and channel number from a channel list 130b in order to send message 122 to a responder 102' operating with a responder configuration 131.

Figure 1e

Figure 1e is an illustration of an exemplary responder bootstrap public keys and initiator bootstrap PKI keys recorded by a plurality of devices, in accordance with exemplary embodiments. Figure 1e depicts PKI keys recorded for an exemplary three different devices 106, although a system 100 and other systems herein could operate with potentially millions or more devices 106. The data depicted for each device in Figure 1e can comprise exemplary data for a responder public key table 101ta and an initiator key table 101tb within an initiator key database 101t, which were depicted and described in connection with Figure 1d above.

The first two entries for responder bootstrap public keys Br 102a for device 106 (1) and device 106(2) in Figure 1e depicts the same alphanumeric values for base85 binary to text encoding for an exemplary responder bootstrap public keys Br 102a in a first device 106(1) and a second device 106(1). Likewise, the second two entries for responder bootstrap public key Br 102a for device 106 (1) and device 106(2) in Figure 1e depicts the same alphanumeric values for base85 binary to text encoding for an exemplary second responder bootstrap public keys Br 102a in first device 106(1) and a second device 106(1). The depiction of these keys Br 102a illustrates the use of shared keys 102z for a plurality of different devices 106. Although only two devices are depicted with shared keys 102z, many more devices could also record the same shared keys for Br 102a. Each of the shared keys 102z is associated with a different network 107, identified with an exemplary network name 107a. In this manner, a plurality of different devices 106 can record and use the same value for a responder bootstrap public key Br 102a.

The same values for shared keys 102z across different devices 106 could be recorded in device 106 during manufacturing or before distribution to end users of device 106. In this manner, devices 106 could be received by end users in a "partially configured" yet secure state, such that a device 106 could use the recorded keys Br 102a with a responder proxy 102', where a responder proxy 102' does not operate or record the corresponding responder bootstrap private key br 102b. As depicted and described in connection with Figures 3a, 3b, 5a, and 5b below, a DPP server 103 could store and

operate with the corresponding responder bootstrap private key br 102b and thus the key br 102b can remain secured and not distributed out or sent to a responder proxy 102'. In this manner, a DPP 191 can be conducted between initiator 101 and a responder proxy 102' securely, where responder proxy 102' could operate on a device with a lower level of security or trust than DPP server 103. In some exemplary embodiments, responder proxy 102' could operate within an insecure device. Although not depicted in Figure 1e or Figure 1d, an entry for a key Br 102a could also record a "trust level" or "privilege level", such that a configuration object received from a DPP 191 using the key Br 102a with the corresponding privilege level could be authorized by device 106 to perform tasks or operations or record data with a higher level of privilege or trust.

In addition, the use of the same values for shared keys 102z across different devices 106 supports simplified yet secure automatic configuration of devices using a DPP 191, where WiFi access points 105 could operate as responder proxies 102'. As depicted and described in connection with Figure 1f below, a WiFi router operating as a responder proxy 102' could potentially connect with at least one network 107 and DPP server 103 (or perhaps a plurality of DPP servers 103 for different networks 107). Responder proxy 102' could use the hash value 102aa for the responder bootstrap public key Br 102a to determine a network 107 identified by the hash value 102aa (see responder network tables 102t below in Figure 1f). In this manner, responder proxy 102' could route data from a DPP authentication request message 122 to the correct DPP server 103 which operates and records (or can obtain from a device database 104 for network 107) the corresponding responder bootstrap private key br 102b in order to conduct the full series of messages for a device provisioning protocol 191.

In exemplary embodiments, responder bootstrap public keys Br 102a can also comprise a unique key such as a unique key 102vv recorded in a tag 102v for responder 102. Thus exemplary embodiment also support the use of a responder bootstrap public key Br 102a that is not shared across multiple initiators 101. In exemplary embodiments, and as depicted in Figure 1e, the initiator bootstrap private key bi 101b and corresponding initiator bootstrap public key Bi 101a can be different for different devices 106. In this manner, a device 106 can also be identified by the hash value 101aa (depicted in Figure 1d) over the initiator bootstrap public key Bi 101a by responder 102' or network 107. As depicted in Figure 1e, an initiator 101 can record a different initiator bootstrap private key bi 101b and corresponding initiator bootstrap public key Bi 101a for each set of supported parameters 199a. In exemplary embodiments, the initiator bootstrap private key bi 101b and corresponding initiator bootstrap public key Bi 101a can be recorded in a secure nonvolatile memory within device 106 for initiator 101 upon device manufacturing or during device distribution.

Figure 1f

Figure 1f is a graphical illustration of hardware, firmware, and software components for a responder, including a responder configuration step, in accordance with exemplary embodiments.

Figure 1f is illustrated to include several components that can be common within a responder 102. Responder 102 may consist of multiple electrical components in order to both (i) communicate with a network 107 and (ii) transmit and receive messages with an initiator 101 according to a device provisioning protocol 191. Responder 102 can undergo a responder configuration step 127 in order to configure the components in a responder 102 to function as the enhanced responder or responder proxy as contemplated in the present disclosure, such that a responder bootstrap private key br 102b can remain recorded with a network 107 as depicted in Figure 1a above. In addition, although a responder 102 in Figure 1a is depicted as a separate component in a network access point 105, a network access point 105 could function and operate with the components of a responder 102 or responder 102' in Figure 1f. In other words, a network access point 105 or another wireless computing device with a WiFi radio 102x could comprise a responder 102 or responder 102' as contemplated herein. Responder 102 and configured responder 102' could also operate with a radio 102x that supports wireless networking technology other than WiFi, such as radio 102x comprising a radio for 4G LTE or 5G networks, and other possibilities exist as well.

In exemplary embodiments and as depicted in Figure 1f, responder 102 can include a responder identity 102i, a processor 102c (depicted as "CPU 102c"), random access memory (RAM) 102d, an operating system (OS) 102e, storage memory 102f (depicted as "nonvolatile memory 102f"), a WiFi radio 102x, a system bus 102j, and a wireless wide area network (WAN) interface 102h. Figure 1d depicts responder 102 as both a default responder 102 and a configured responder 102'. The default responder 102 can comprise the state of responder 102 before a configuration step 127 and responder 102' can comprise the state of responder 102' after the configuration step 127. As depicted, both a default responder 102 and a configured responder 102' can contain the same hardware components such as CPU 102c, RAM 102d, etc., where a difference between responder 102 and 102' can be in both (i) data recorded in storage memory 102f and a change in the operation of WiFi radio 102x to listen for incoming DPP messages 122.

Responder identity 102i could comprise a preferably unique alpha-numeric or hexadecimal identifier for responder 102, such as an Ethernet or WiFi MAC address, an International Mobile Equipment Identity (IMEI), an owner interface identifier in an IPv6 network, a serial number, or other sequence of digits to uniquely identify each of the many different possible units for responder 102 for a network 107. A system 100 depicted in Figure 1a could operate with a plurality of different responders 102. Responder identity 102i can preferably be recorded in a non-volatile memory or written directly to hardware in responder 102 by (i) a responder manufacturer upon device

manufacturing, or (ii) a pre-configuration step for responder 102 before starting the sequence of steps and message flows depicted in Figure 3a below.

The CPU 102c can comprise a general purpose processor appropriate for typically low power consumption requirements for a responder 102, and may also function as a microcontroller. CPU 102c can comprise a processor for responder 102 such as an ARM® based processor or an Intel® based processor such as belonging to the Atom or MIPS family of processors, and other possibilities exist as well. CPU 102c can utilize bus 102j to fetch instructions from RAM 102d and operate on the instruction. CPU 102c can include components such as registers, accumulators, and logic elements to add, subtract, multiply, and divide numerical values and record the results in RAM 102d or storage memory 102f, and also write the values to an external interface such as WiFi radio 102x or WAN interface 102h.

RAM 102d may comprise a random access memory for responder 102. RAM 102d can be a volatile memory providing rapid read/write memory access to CPU 102c. RAM 102d could be located on a separate integrated circuit in responder 102 or located within CPU 102c. The RAM 102d can include data recorded in responder 102 for the operation when collecting and communicating DPP messages 122 through 124 with initiator 101 depicted in Figures 1b, 1c, 3a, etc. The system bus 102j may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures including a data bus. System bus 102j connects components within responder 102 as illustrated in Figure 1f, such as transferring electrical signals between the components illustrated. Responder 102 can include multiple different versions of bus 102j to connect different components, including a first memory bus 102j between CPU 102c and RAM 102d, and a second system bus 102j between CPU 102c and WiFi radio 102x, which could be an I2C bus, an SPI bus, PCIe bus, or similar data busses.

The operating system (OS) 102e can include Internet protocol stacks such as a User Datagram Protocol (UDP) stack, Transmission Control Protocol (TCP) stack, a domain name system (DNS) stack, a TLS stack, a datagram transport layer security (DTLS) stack, etc. The operating system 102e may include timers and schedulers for managing the access of software/firmware to hardware resources within responder 102, including CPU 102c and WiFi radio 102x. The operating system shown of 102e can be appropriate for a low-power device with more limited memory and CPU resources (compared to a server such as DPP server 103, device database 104, etc.). Example operating systems 102e for a responder 102 includes Linux, Android® from Google®, IOS from Apple®, Windows® 10 IoT Core, or Open AT® from Sierra Wireless®. Additional example operating systems 102e for a responder 102 include eCos, uC/OS, LiteOs, Contiki, OpenWRT, Raspbian, and other possibilities exist as well without departing from the scope of the present disclosure. Although depicted as a separate element within responder 102 in Figure 1f, OS 102e may reside in RAM 102d and/or storage memory 102f during operation of responder 102.

Storage memory 102f (or “nonvolatile memory 102f”) within responder 102 can comprise a nonvolatile memory for storage of data when responder 102 is powered off. Memory 102f may be a NAND flash memory or a NOR flash memory, and other possibilities exist as well without departing from the scope of the present disclosure. Memory 102f can record firmware and/or software for responder 102. Memory 102f can record long-term and non-volatile storage of data or files for responder 102. In an exemplary embodiment, OS 102e is recorded in memory 102f when responder 102 is powered off, and portions of memory 102f are moved by CPU 102c into RAM 102d when responder 102 powers on. Memory 102f (i) can be integrated with CPU 102b into a single integrated circuit (potentially as a “system on a chip”), or (ii) operate as a separate integrated circuit or a removable card or device, such as a removable SD card. Memory 102f may also be referred to as “device storage” and can include exemplary file systems of FAT16, FAT 32, NTFS, ext3, ext4, UDF, or similar file systems.

As depicted in Figure 1f, nonvolatile memory 102f for a default responder 102 may also contain a secret key SK0.responder 102s, a certificate cert.responder 102m, and a certificate cert.CA.root 102w. A secret key SK0.responder 102s in a nonvolatile memory 102f could comprise the private key for a PKI key pair, where the corresponding public key could be recorded in a certificate cert.responder 102m. A secret key SK0.responder 102s and the corresponding public key in a certificate 102m could comprise values to use with asymmetric public key infrastructure (PKI) cryptography, and could utilize exemplary algorithms based on either Rivest Shamir Adleman (RSA) or elliptic curve cryptography (ECC). For use of ECC algorithms, parameters within certificate 102m (and other certificates herein) can specify elliptic curve names (e.g. NIST P-256, sect283k1, sect283r1, sect409k1, sect409r1, etc.). The same ECC parameters can be used with bootstrap PKI keys in the form of parameters 199a. Further, elliptic curves that do not depend on parameters specified by NIST could be utilized as well, such as Curve22519 or FourQ. For use of RSA algorithms, parameters within certificate 102m can specify a modulus and other associated values for using an RSA PKI key pair. For either asymmetric algorithms RSA or ECC for a PKI key pair herein, parameters in a certificate such as cert.responder 102m can specify key lengths, a duration for key validity, uses of the PKI key pair such as for key derivation or signatures, encoding rules, message digest algorithms, etc. Parameters in certificate 102m (and other certificates herein) may also include identifying information associated with the PKI key pair such as a sequence number, a serial number, a certificate revocation list or URL, and a domain name of the server associated with the PKI key pair.

Certificate cert.CA.root 102w can comprise a root certificate for responder 102 to authenticate a server certificate received from DPP server 103 or other servers or nodes in a network 107, where these other nodes for network 107 are depicted and described in connection with Figure 1a, Figure 3a, and other Figures below. After authenticating a server certificate received by responder 102 or 102' with cert.CA.root 102w, responder 102 can conduct additional authentication steps contemplated by

secure protocol standards such as Transport Layer Security (TLS), Datagram Transport Layer Security (DTLS), IPsec, etc. Root certificate cert.CA root 102w can comprise the list or a subset of the list of included root certificates from the Mozilla Foundation with Mozilla projects, where the aggregate list of community approved root certificates and associated parameters is in the widely distributed file certdata.txt from the Mozilla web site.

Nonvolatile memory 102f in a configured responder proxy 102' can include a responder network table 102t. Responder network table 102t could be received from a network 107 for a configuration step 127 in a message 305 depicted below in Figure 3a. Or, table 102t could be loaded by a user of responder 102' or potentially recorded during a pre-configuration of responder 102 or responder 102' before the sequence of message flows and steps in Figure 3a below. Responder network table 102t can record and map a DPP server 103 that is associated with a responder bootstrap public key hash value 102aa. The responder bootstrap public key hash value 102aa can be associated with a network 102ab for responder bootstrap public key Br 102a (e.g. the network equivalent to network 107 that records the responder bootstrap private key br 102b to process an incoming DPP authentication request message 122). The configured responder proxy 102' can use the responder network table 102t to route data from incoming DPP messages 122 with the responder bootstrap public key hash value 102aa to a selected DPP server 103. Different DPP servers 103 could be associated with different networks 107, where a network 107 can comprise a network for H(Br) 102aa, as depicted for a responder network table 102t in Figure 1f. Although not depicted in Figure 1f, a responder network table 102t could also include a set of cryptographic parameters 199a associated with the responder bootstrap public key hash value 102aa, including the hash algorithm 199e associated with the responder bootstrap public key hash value 102aa.

In some exemplary embodiments, the responder network table 102t with a plurality of responder bootstrap public key hash values 102aa could be omitted from a memory of responder 102, and a DPP server 103 or network 107 could record the values. A network 107 or DPP server 103 could use a hash value 102aa received in a message 309 (depicted and described below in connection with Figure 3a) in order to select a DPP server 103. In other exemplary embodiments, a responder network table 102t could include at least one value for a responder bootstrap public key hash value 102aa.

In other words, a plurality of different devices 106 with initiators 101 may use different responder bootstrap public keys Br 102a, where the hash value 102aa would be received in an incoming DPP message 122. The devices 106 could be manufactured or pre-configured with at least one key Br 102a, and the manufacturer could operate or be associated with a DPP server 103 in order to support and conduct a DPP 191 with device 106 and send a configuration object in a DPP configuration response message 124 from Figure 1c. Note that a configuration object in a message 124 could include data other than credentials 109 (and may optionally omit credentials 109), where

the configuration object could include configuration parameters for device 106. Responder 102' can use the received hash value 102aa to select a DPP server 103 in order to further process the message and support a DPP 191, where the network 102ab could be different than a network 107 operating a network access point 105. In exemplary embodiments a network 102ab operating a DPP server with responder bootstrap private key br 102b can be selected using the hash value 102aa over key responder bootstrap public key Br 102a.

A device 106 with initiator 101 could use (i) a first DPP 191 with a first Br 102a to obtain network credentials 109 with network 107 for AP 105, and (ii) a second DPP 191 with a second Br 102a (with the same responder proxy 102') to obtain configuration parameters with network 102ab for device 106. The first DPP 191 could be with a first DPP server 103 and first network 102ab, and the second DPP 191 with a second DPP server 103 and a second network 102ab. The use of responder network table 102t by responder 102' can determine which incoming DPP message 122 is associated with which network 102ab and DPP server 103. In other words, a responder 102' can conduct multiple sessions of DPP 191 with different keys Br 102a, and use a responder network table 102t to determine which DPP messages 122 are associated with which DPP server 103 (via hash value 102aa). Other possibilities exist as well for a responder 102' to use a responder network table 102t in order to process different incoming DPP authentication request messages 122 without departing from the scope of the present disclosure. In exemplary embodiments, a selected DPP server 103 from a responder network table 102t can provide credentials 109 for a DPP configuration response 124, using the steps below such as those depicted in Figure 3a, 3b, Figure 6, etc.

Responder 102 and 102' can include a WiFi radio 102x to communicate wirelessly with networks such as device 101 depicted and described in Figure 1a above, including via WiFi data link 192. WiFi radio 102x could connect with an antenna in order to transmit and receive radio frequency signals. Responder 102 and 102' can operate a WAN interface 102h in order to communicate with network 107, and WAN interface can include a WAN MAC address 102q. WAN interface 102h can comprise either a wireless connection, such as with a 4G LTE network, a 5G network, or can comprise a wired connection such as digital subscriber line (DSL), coaxial cable connection, or Ethernet connections, and other possibilities exist as well without departing from the scope of the present disclosure.

As depicted in Figure 1f, a WiFi radio 102x in a configured responder 102' can record or operate with a responder configuration 131. The values for responder DPP configuration 131 could also specify a version of the 802.11 standards to utilize, such as, but not limited to, 802.11n, 802.11ac, 802.11ah, 802.11ax, or related and subsequent versions of these standards. The responder DPP configuration 131 can be compatible with the initiator DPP configuration 130 of a target WiFi device 101 with responder 102'. The responder configuration 131 could be obtained during an authentication step for responder 102, such as depicted below in Figure 3a (in a message 305), although responder

configuration 131 could be obtained at other times as well, such as possibly being recorded in a WiFi radio 102x before responder 102 begins sending and receiving data through a WiFi link 192.

The responder DPP configuration 131 can include a responder operating class 131a which can be selected from the plurality of potential global WiFi operating classes in table E-4 of IEEE standard specification 802.11ac, which is incorporated by reference herein. The present disclosure contemplates that a responder 102' communicates with multiple initiators 101 either concurrently, or over time. The responder configuration 131 can also include Responder DPP Channel Authentication List 131b, which can comprise a list of channels for responder 102' to receive DPP authentication request 122 messages in order to establish initial connectivity with device 106 and initiator 101. In exemplary embodiments, responder configuration 131 for responder 102' provides data for a responder 102 to receive data such as message 122 from initiator 101, where initiator 101 may operate with different possible values for operating classes or transmitting channels.

A responder configuration 131 can include other data for responder 102' to receive a DPP message from an initiator 101 as well, such as specifying the use of broadcast or unicast for receiving DPP messages, timer values for retry of sending DPP 191 messages, timer values for timeouts for sending and receiving messages, as well as counters for a maximum number of retries on sending messages. In other words, without responder configuration 131, a responder 102 may not normally be able to (i) select the proper WiFi operating channel, and (ii) receive broadcast data from initiator 101. For embodiments where a Device DPP Channel Authentication List 131b is unknown or not available, then responder 102' can periodically listen for incoming DPP 122 messages over time as broadcast messages on all available channels for a radio 102x.

WiFi radio 102x can include standard radio components such as RF filters, RF amplifiers, a clock, and phased loop logic (PLL). WiFi radio 102x can support protocols and specifications according to the IEEE 802.11 family of standards. For example, if WiFi radio 102x operates according to 802.11n standards, WiFi radio 102i could operate at a radio frequency of 2.4 or 5 GHz. In other embodiments where WiFi radio 102i supports 802.11ac standards, then WiFi radio 102x could also support radio frequencies of 0.054 through 0.79 GHz in order to operate in "white space" spectrum. Other possibilities exist as well for WiFi radio to support wireless LAN standards or 802.11 standards and radio frequencies, without departing from the scope of the present disclosure. WiFi radio 102x can access a nonvolatile memory 102f in order to record the depicted configuration values of responder configuration 131 where the nonvolatile memory could be within WiFi radio 102x, or WiFi radio 102x could access memory 102f via bus 102j in order to record and read the values. Device 106 can operate a WiFi radio 106r that is compatible with WiFi radio 102x.

As depicted in Figure 1f, a WiFi configured responder 102' can include configuration data resulting from a configuration step 127, in order for responder 102' to conduct a device provisioning protocol 191 with a device 101. A nonvolatile memory 102f for a WiFi configured responder 102'

can include data recorded for a default responder 102, with the addition of recording a DPP application program 102g. DPP application program 102g (or “DPP application 102g”) could comprise a software package for OS 102e in order for responder 102’ to conduct the series of steps and messages with DPP server 103 and initiator 101 depicted below in Figure 3a, Figure 3b, Figure 4a, etc. DPP application 102g can include cryptographic algorithms in order to support key exchanges, symmetric encryption and decryption, as well as signature verification and asymmetric encryption and decryption. As one example, a DPP application 102g could perform the function of a key derivation 102y from Figure 1b, and other possibilities exist as well.

Responder 102 may also optionally include user interface 102k which may include one or more devices for receiving inputs and/or one or more devices for conveying outputs. User interfaces are known in the art and may be simple for many devices such as a few LED lights or and LCD display, and thus user interfaces are not described in detail here. User interface 102k could comprise a touch screen if responder 102 has sophisticated interaction with user 102a. Responder 102 can optionally omit a user interface 102k, if no user input or display is required for conducting a device provisioning protocol 191 between responder 102 and device 106 with initiator 101. Although not depicted in Figure 1f, responder 102 can include other components to support operation, such as a clock, power source or connection, antennas, etc. User interface 102k could also comprise a web server operating in responder 102 and responder 102’, such that a user could access the user interface 102k with a web browser and view or control depicted data stored in memory for a responder 102 or configured responder 102’. Other possibilities exist as well for hardware and electrical components operating in responder 102 or responder 102’ without departing from the scope of the present disclosure.

Although not depicted in Figure 1f, the various servers shown above in Figure 1a such as DPP server 103 and device database 104 can include equivalent internal electrical components depicted for a responder 102 in order to operate as servers. The servers in Figure 1a could include a processor similar to CPU 102c, with primary differences for the processor in a server being increased speed, increased memory cache, an increased number and size of registers, with the use of a 64 bits for datapath widths, integer sizes, and memory address widths, etc., In contrast, an exemplary 32 bit datapaths could be used for CPU 102c in responder 102 (although CPU 102c could also use 64 bit wide datapath widths if responder 102 comprises a mobile phone such as a smart phone). For embodiments where responder 102 comprises a home WiFi router, then a CPU 102c in responder 102 could comprise an exemplary 32 bit processor, although other possibilities exist as well.

Similarly, RAM in a server could be a RAM similar to RAM 102d in responder 102, except the RAM in a server could have more memory cells such as supporting exemplary values greater than an exemplary 8 gigabytes, while RAM 102d in responder 102 could support fewer memory cells such as less than an exemplary 4 gigabytes. Non-volatile memory for storage in a server in Figure 1a could comprise disks, “solid state drives” (SSDs) or “storage area networks” (SAN) for a server. For a WAN

interface in a server equivalent to WAN interface 102h for responder 102', in exemplary embodiments, a server in Figure 1a could use a physical, wired LAN interface such as a high-speed Ethernet or fiber optic connection. In exemplary embodiments, a device 106 with an initiator 101 can include and operate with hardware and electrical components similar or equivalent to the components for a responder 102' depicted in Figure 1f.

Figure 2a

Figure 2a is an illustration of an exemplary device database, with tables for a device database recording exemplary data, in accordance with exemplary embodiments. A device database 104 depicted and described above in connection with Figure 1a can record data for (i) an unconfigured responder 102 or (ii) a configured responder 102' and a DPP server 103 to conduct a device provisioning protocol 191 with a device 106 operating an initiator 101. A database 104 could record a plurality of tables for a plurality of devices 106 and/or responders 102', such as the depicted values for a key table 104k, and a set of cryptographic parameters 199. Other possibilities exist as well for the organization, tables, and recorded data within a device database 104 without departing from the scope of the present disclosure. Data within device database 104 could be encrypted using a symmetric key. As depicted in Figure 1a, a network 107 could operate and record a device database 104, although another entity besides network 107 could also operate and record the device database 104, such as a device manufacturer or a device owner. For embodiments where a plurality of different networks 107 may communicate with a device 106 through responder 102' over time, each of the different networks 107 can operate a device database 104.

Device database 104 can record sets of ID.device 106i, MAC.I 106n for the WiFi MAC address of the device 106 operating the initiator 101, responder identity 102i, a responder mode 306a, cryptographic parameters 199, and a secret shared key 198 for a public key exchange protocol (PKEX). These values within a device database could be recorded in a device 106 and/or initiator 101 upon or after manufacturing of a device 106 or distribution of device 106 and before device 106 conducts a DPP 191 with responder 102'. The exemplary data in device database 104 could be obtained from the manufacturer or device distributor. For embodiments where some data is not previously recorded in a device database 104 before an initiator 101 sends a message 122 as depicted below in Figure 3a, then a network 107 operating device database 104 could query the device 106 manufacturer or a device 106 distributor in order to receive and record values for a device database 104, and other possibilities exist as well for the source of device 106 data within a device database 104. Note that key 198 for PKEX can be optionally omitted for some devices, with the depiction of "NA".

ID.device 106i can comprise an identity for device 106i, and could be a serial number or identification number or string for device 106. The two identity values shown for a device 101 in a

device database 104 allows a DPP server 103 or network 107 to map or convert between ID.device 106i and MAC.I 106n, and also convert one value into another. MAC.I 106n can comprise the MAC address used by an initiator 101 for conducting a device provisioning protocol 191. In other words, MAC.I 106n can be used by a responder 102' or network 107 in order to identify that an incoming DPP authentication request message 122 is from device 106 (and not a fraudulent or imposter device 106). For embodiments where key Br 102a comprises a shared key 102z, MAC.I 106n can also be used by network 107, database 104, DPP server 103, or responder 102' to verify the message 122 is from device 106 (since only device 106 with MAC.I 106n would normally send the message 122). A message 122 would have a source MAC address, which could be checked against MAC.I 106n that was recorded in a device database 104. Further, the hash value 101aa from a message 122 could be checked against a hash value calculated by device database 104 or DPP server 103 over the recorded initiator bootstrap public key Bi 101a. Additional information for verifying an identity for device 106 is provided for a step 312 below in Figure 3a.

In some exemplary embodiments, MAC.I 106n can be omitted from a device database 104 or otherwise not available to a network 107, and in this case the MAC.I 106n can be recorded in a database 104 after successful receipt of a DPP authentication confirm message 123a depicted in Figure 3b below. In other words, some devices 101 with initiators 106 may change or randomize the MAC.I 106n over time, and the MAC.I 106n stored in a device database 104 can comprise the value of MAC.I 106n used by device 106 to send a DPP authentication request message 122. A network 107 with a device database 104 could operate a plurality of different responders 102'. The responder identity 102i of a responder 102' communicating with a device 106 could be recorded in a device database 104, although in some exemplary embodiments the responder identity 102i could be optionally omitted. Responder identity 102i could be omitted or blank in a device database 104 until after a message 122 is received by a responder 102' and forwarded to device database 104 via network 107.

A device database 104 could also specify a responder mode 306a (in Figure 3a and Figure 3c below) for responder 102' to operate with initiator 101, where different responder modes could comprise a first responder mode as depicted in Figure 3a, where DPP server 103 records and operates on (a) responder bootstrap private key br 102b and (b) ephemeral PKI keys for responder 102' and (c) all derived secret shared keys (as depicted in the first row of Figure 3c). A second responder mode 306a could comprise DPP server 103 recording and operating on responder bootstrap private key br 102b, but not derived responder ephemerals private key pr 102r for responder 102' (e.g. as depicted in Figure 5a). A third responder mode 306a could comprise DPP server 103 recording and operating on responder bootstrap private key br 102b and derived responder ephemeral private key pr 102r, but not derived secret shared keys for encryption such as keys k1, k2, and ke (e.g. as depicted in Figure 5b).

In some exemplary embodiments, a device 106 could comprise a higher value device 106 or require a greater level of security, so the security level of a DPP 191 could be selected using a responder mode 306a. In addition, the responder mode 306a could be selected based on the security level of a responder 102', such that a first responder mode 306a such as in Figure 3a and Figure 3b is selected by network 107 for a responder 102' with a lower level of security. A second responder mode 306a such as depicted in Figure 5b is selected by a network for a responder 102' with a higher level of security, such as a gnodeb in a 5G network or a secured enterprise WiFi router. Other reasons for the selection of a responder mode exist as well without departing from the scope of the present disclosure.

Sets of cryptographic parameters 199 in device database 104 can represent different sets of cryptographic parameters supported by responder 102' and/or initiator 106 in order to conduct a device provisioning protocol 191. For example, some devices 106 may use a first set of cryptographic parameters with PKI keys such as bootstrap and ephemeral keys, and other devices may use a second set of cryptographic parameters, which could be specified by cryptographic parameters 199. Note that devices 101 can optionally support more than one set of cryptographic parameters, as depicted for device with exemplary identity ID.device 106i of "20003", where the device can operate with either a set "A" of cryptographic parameters 199 or a set "B" of cryptographic parameters.

PKEX key 198 could be used with a public key exchange protocol as specified in IETF draft document "Public Key Exchange", which is hereby incorporated by reference in its entirety. In some exemplary embodiments, a PKEX key 198 can be omitted, such as for the example data depicted with an "NA" in Figure 2b. The values depicted of X1, X2, etc. for a PKEX key 198 could comprise the secret key for each device 106 identified by MAC.I 106n. For the exemplary database 104 depicted in Figure 2a and other tables herein, a numeric value following a constant name can depict a subsequent or different instance or numeric value for the constant. In other words, "X1" and "X2" for a PKEX key 198 can represent different values of "X" for the PKEX. PKEX key 198 could be used as depicted and described in Figure 3a with a PKEX session 312e in order to query device 106 for the initiator bootstrap public key Bi 101a if a database 104 does not record the key Bi 101a (or if network 107 cannot obtain key Bi 101a via other means via an IP network 113).

Device database 104 can record a key table 104k, which could comprise exemplary keys as depicted. A key table 104k can include responder bootstrap public key Br 102a, responder bootstrap private key br 102b, initiator bootstrap public key Bi 101a, a secure hash value for responder bootstrap public key 102aa, and a secure hash value for initiator bootstrap public key 101aa. Similarly, values of "Br-1" and "Br-2" depict different values for a responder bootstrap public key 102a, but exemplary values of "Br-1" and then a second "Br-1" can depict that the same value or number may be recorded as the responder bootstrap public key Br 102a in different devices 106 each with an initiator 101. The use of the same value for a responder bootstrap public key Br 102a across different devices 106 can

comprise a shared responder bootstrap public key 102z, which is also depicted and described in connection with Figure 1e above.

Responder bootstrap public key Br 102a in a device database 104 can represent the bootstrap public key for responder 102'. Br 102a was also depicted and described above in connection with Figure 1a, Figure 1b, and Figure 1e above. In exemplary embodiments, a device database 104 can record a plurality of different initiator bootstrap public keys Bi 101a, where each key for a set of parameters 199 can be associated with different initiators 101. As depicted in Figure 2b, for some embodiments different initiators 101 could also share the same numeric value for an initiator bootstrap public key Bi 101a (e.g. depicted devices "20001" and "20002" can both use an initiator bootstrap public key Bi 101a of "Bi-1"). Also note that a DPP 191 can be conducted between a responder 102' and initiator 101 without recording an initiator bootstrap public key Bi 101a in a device database 104 or DPP server 103 or responder 102', which could comprise a responder only authentication 141 as depicted in Figure 1b. For these embodiments, the depicted value for Bi 101a can be "NA" as shown in Figure 2a.

In a key table 104k in device 104, different initiator bootstrap public keys Bi 101a can be associated with a different selected set of parameters 199a. The manufacturer or entity configuring initiator 101 in device 106 may not know which of a plurality of different possible sets of cryptographic parameters 199 may be utilized by an initiator 101 for conducting a DPP 191 in order to be compatible with a responder 102'. For example, DPP specification version 1.0 identifies several possible sets of cryptographic parameters 199 for bootstrap keys that could be utilized in a DPP 191, such as the different values for cryptographic parameters shown in Appendix A of DPP specification version 1.0. As depicted in Figure 2a, an initiator bootstrap public key Bi 101a of "Bi-5 (A)" can represent an instance or number for Bi 101a which uses the selected set "A" of cryptographic parameters 199 (e.g. curve p256 with SHA-256), while initiator bootstrap public key Bi 101a of "Bi-5 (D)" can represent an instance of Bi 101a which uses the selected set "D" of cryptographic parameters 199 (e.g. curve 25519 with SHA-3). Similar designations and uses of different cryptographic parameters 199 with different responder PKI keys can be recorded in a device database 104 as well. In some exemplary embodiments, the values depicted for responder keys could be omitted from a device database 104 and recorded in another server such as DPP server 103.

Responder bootstrap public key Br 102a and corresponding responder bootstrap private key br 102b in a device database 104 can represent the bootstrap public key and the bootstrap private key for a responder proxy 102'. Br 102a and br 102b were depicted and described above in connection with Figures 1a through Figure 1e above. In exemplary embodiments, a device provisioning protocol 191 using a responder 102' and a DPP server 103 can utilize a plurality of different responder bootstrap PKI keys, and different values for the responder bootstrap PKI keys Br 102a and br 102b can be associated with different initiators 101. As discussed above in connection with Figure 1a and 1b, an

initiator 101 could record or obtain a responder bootstrap public key Br 102a, and subsequently the value Br 102 recorded for the initiator 101 could also be stored in a device database 104.

The responder bootstrap public key Br 102a and corresponding responder bootstrap private key br 102b could be stored by a device database 104 concurrently with the initiator bootstrap public key Bi 101a, where the initiator bootstrap public key Bi 101a could be (i) received from a device manufacturer or device distributor, or (ii) obtained by a PKEX session 312e (in Figure 3a), or (iii) responder 102' scanning a QR code or bar code for initiator 101 (and sending the key Bi 101a to device database 104), and other possibilities exist as well for the source of key Bi 101a in a database 104 for device 106 with MAC.I 106n. Or, in exemplary embodiments a device manufacturer or a device distributor could operate a device database 104 and subsequently record the exemplary keys and values depicted of (i) Bi 101a for initiator 101 in device 106 and (ii) recorded Br 102a in device 106, but not (iii) br 102b, which could be recorded by a device database 104 for a network 107 operating or associated with responder 102'. As depicted in Figure 2a, some responders 102' could share a common responder bootstrap public key Br 102a, which could be associated with the same responder bootstrap private key br 102b.

The sharing of responder bootstrap public key Br 102a among multiple devices 106 with initiators 101 (e.g. "shared key 102z") can simplify the manufacturing and distribution of devices by using a common responder bootstrap public key Br 102a for some embodiments. For the present disclosure, the potential tradeoff of greater simplicity for reduced security (by reusing a responder bootstrap public key Br 102a in multiple devices 106) can be avoided, since the responder bootstrap private key br 102b may (i) preferably not be sent to the responder 102', and (ii) remain recorded in and operated by a secured DPP server 103.

Yet, with the embodiments contemplated herein, the full set of device provisioning protocol messages 191 can be sent and received with an initiator 101 by responder proxy 102'. The full set of DPP 191 messages can be transmitted and received by a responder 102' with initiator 101 using either responder authentication 141 or mutual authentication 142. Embodiments for the present disclosure depict and describe the responder bootstrap private key br 102b as residing within a DPP server 103 when a responder 102' conducts a DPP 191 (as depicted and described in connection with Figure 3a, Figure 3b, Figure 5a, etc. below). Consequently in exemplary embodiments and a difference with conventional technology depicted in Figure 1b and Figure 1c, (i) the responder bootstrap private key br 102a does not need to be recorded by a responder 102' (but br 102a is recorded for a conventional responder 102* in Figure 1b and Figure 1c), and (ii) a responder bootstrap public key Br 102a can be commonly shared between multiple initiators 101 in different devices 106 (e.g. "shared key 102z").

Thus, the present disclosure resolves the potential tradeoff of security versus simplification by supporting manufacturing and distribution of devices 106 with initiators 101 with commonly shared responder bootstrap public key Br 102a (also depicted in Figure 1e as "shared key 102z"). Additional

details will be provided in Figures below. The responder bootstrap private key br 102b can be recorded in a device database 104 and subsequently sent to a DPP server 103, but preferably not sent to responder 102', and responder 102' can still conduct a device provisioning protocol 191 for initiator 101 using mutual authentication 142 as depicted in Figure 1c. In exemplary embodiments, a responder bootstrap public key Br 102a is recorded inside an initiator 102, while a device database 104 can record both the responder bootstrap public key Br 102a and the responder bootstrap private key br 102b, as depicted in Figure 2a.

Cryptographic parameters 199 can specify sets of cryptographic parameters that are supported by responder 102' when using bootstrap PKI keys. Cryptographic parameters 199 can be recorded in a device database 104. Cryptographic parameters 199 can record a collection of cryptographic algorithms or specifications such as a set identifier 199a, a key length 199b, an ECC curve name 199c, a hash algorithm 199e, a key length for a symmetric ciphering algorithm 199f, and a nonce length 199g. Asymmetric encryption parameters 199h can specify values and parameters to use with an asymmetric encryption algorithm 351a depicted and described in connection with Figure 3e below. Parameters 199i can specify encoding rules, padding, key lengths, selected algorithms, curve names, use of deterministic mode, and other values or fields necessary to utilize a signature algorithm 338b depicted and described in connection with Figure 3d below. In exemplary embodiments, parameters 199i can specify ASN.1 syntax notation (abstract syntax notation) with distinguished encoding rules (DER). Parameters 199i can also specify the use of either (i) ephemeral PKI keys or (ii) bootstrap PKI keys for use with a digital signature algorithm.

As contemplated herein, when a selected set of cryptographic parameters such as using the words or description "parameters 199a" or "cryptographic parameters 199a" can specify a row of parameters or values in a set of cryptographic parameters 199, such that the collection of values in the row can be used with a device provisioning protocol 191. Set identifier 199a can be an identity for a row or set of values for cryptographic parameters 199. For example, set "A" can comprise cryptographic suite 1 as specified in section 3.2.3 of DPP specification version 1.0. Key length 199b can be the length of keys in bits for PKI keys used with a device provisioning protocol with an initiator 101 in a device 106. ECC Curve name 199b can be a name for an ECC curve used with PKI keys for a device provisioning protocol with an initiator 101 in a device 106.

Hash algorithm 199e in cryptographic parameters 199 can be the name of a secure hash algorithm, such as the exemplary SHA-256 algorithm depicted, which may also be referred to as "SHA-2". Hash algorithm 199e can be used in a key derivation function (e.g. 402a, 402b, etc. below) and also to generate a signature based on a received nonce. A key length for a symmetric ciphering algorithm 199f can specify the length of a symmetric key in bits to use with a symmetric ciphering algorithm. The depicted algorithm in Figure 2a for algorithm 199f is "AES-SIV", although other symmetric ciphering algorithms could be specified in a set of cryptographic parameters as well. In

other words, an additional, different column in cryptographic parameters 199, such as an exemplary “symmetric algorithm” 199z (not depicted) could specify the symmetric ciphering algorithm to utilize with a responder for a device provisioning protocol. Nonce length 199g can specify the length in bits for nonces or random numbers generated by both initiator 102’ and responder 101x, including the length in bits of authentication values, where the nonces can be used for generating signatures and encryption keys.

Figure 2b

Figure 2b is a flow chart illustrating exemplary steps for an initiator to use an initiator DPP configuration and initiator key database in order to automatically connect with a responder proxy, in accordance with exemplary embodiments. An initiator 101 can use the exemplary data recorded in an initiator DPP configuration 130 and initiator key database 101t in order to automatically connect with a responder proxy 102’. The steps depicted in Figure 2b allow a device 106 with initiator 101 to locate or find a compatible responder proxy 102’ in an automated manner, such that no manual intervention is required by a device user to conduct a DPP 191 with subsequent configuration step 106a in order to load a device with credentials 109. This provides benefits over a conventional DPP 191, where additional manual intervention is required for operating a device with initiator 101 as a configurator and obtaining a bootstrap responder public key Br 102a via an “out of band” method, or loading a device 106 with a PKEX key 198.

The processes and operations, described below with respect to all of the logic flow diagrams and flow charts may include the manipulation of signals by a processor and the maintenance of these signals within data structures resident in one or more memory storage devices. For the purposes of this discussion, a process can be generally conceived to be a sequence of computer-executed steps leading to a desired result.

These steps usually require physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is convention for those skilled in the art to refer to representations of these signals as bits, bytes, words, information, elements, symbols, characters, numbers, points, data, entries, objects, images, files, or the like. It should be kept in mind, however, that these and similar terms are associated with appropriate physical quantities for computer operations, and that these terms are merely conventional labels applied to physical quantities that exist within and during operation of the computer.

It should also be understood that manipulations within the computer are often referred to in terms such as listing, creating, adding, calculating, comparing, moving, receiving, determining, configuring, identifying, populating, loading, performing, executing, storing etc. that are often

associated with manual operations performed by a human operator. The operations described herein can be machine operations performed in conjunction with various input provided by a human operator or user that interacts with the device, wherein one function of the device can be a computer.

5 In addition, it should be understood that the programs, processes, methods, etc. described herein are not related or limited to any particular computer or apparatus. Rather, various types of general purpose machines may be used with the following process in accordance with the teachings described herein.

10 The present disclosure may comprise a computer program or hardware or a combination thereof which embodies the functions described herein and illustrated in the appended flow charts. However, it should be apparent that there could be many different ways of implementing the disclosure in computer programming or hardware design, and the disclosure should not be construed as limited to any one set of computer program instructions.

15 Further, a skilled programmer would be able to write such a computer program or identify the appropriate hardware circuits to implement the disclosed disclosure without difficulty based on the flow charts and associated description in the application text, for example. Therefore, disclosure of a particular set of program code instructions or detailed hardware devices is not considered necessary for an adequate understanding of how to make and use the disclosure. The inventive functionality of the claimed computer implemented processes will be explained in more detail in the following description in conjunction with the remaining Figures illustrating other process flows.

20 Further, certain steps in the processes or process flow described in all of the logic flow diagrams below must naturally precede others for the present disclosure to function as described. However, the present disclosure is not limited to the order of the steps described if such order or sequence does not alter the functionality of the present disclosure. That is, it is recognized that some steps may be performed before, after, or in parallel other steps without departing from the scope and
25 spirit of the present disclosure.

30 The processes, operations, and steps performed by the hardware and software described in this document usually include the manipulation of signals by a CPU or remote server and the maintenance of these signals within data structures resident in one or more of the local or remote memory storage devices. Such data structures impose a physical organization upon the collection of data stored within a memory storage device and represent specific electrical or magnetic elements. These symbolic
35 representations are the means used by those skilled in the art of computer programming and computer construction to most effectively convey teachings and discoveries to others skilled in the art.

At step 201, device 106 can be powered on and activate a WiFi radio 106r. A step 201 can include device 106 loading a bootloader program and an operating system for device 106, as well as
35 starting a process or operating firmware to perform the steps of an initiator 101. At step 202, device 106 can load initiator DPP configuration 130 and initiator key database 101t from nonvolatile memory,

where DPP configuration 130 and initiator key database 101t were depicted and described above in connection with Figure 1d.

At step 203, device 106 can select a set of parameters 199 and then all recorded responder bootstrap public keys 102a and at least one initiator bootstrap private key bi 101b for the parameters
 5 199 from an initiator key database 101t. A hash value 101aa for the corresponding initiator bootstrap public keys Bi 101a can be selected or processed in a step 203 as well. In addition, the initiator bootstrap public key Bi 101a can be selected in a step 203.

At step 204, for each set of parameters 199, the initiator 101 in device 106 can generate an ephemeral PKI key pair comprising initiator ephemeral private key pi 101d and corresponding initiator
 10 ephemeral public key Pi 101c. Initiator 101 in device 106 can then process or generate a DPP authentication request message 122 for each responder bootstrap public key Br 102a selected in step 203. As depicted in Figure 3a below, the series of steps for processing a message 122 in a step 204 can comprise steps 313 through 316.

At step 205, initiator 101 operating in device 101 can send (i) a DPP authentication request
 15 message 122 on the current operating WiFi channel 130b within an operating class 130a for (ii) each responder bootstrap public key Br 102a selected in a step 203. For the first instance of step 205, the first WiFi channel 130b and associated operating class 130a can be the default channel 130b and class 130a selected by device 106 for WiFi radio 106r upon power up in step 201. A step 205 can also comprise sending each DPP message 122 for each key Br 102a multiple times with retry timers, such
 20 as waiting a few seconds between sending each DPP message 122. In exemplary embodiments, each DPP message 122 in a step 205 is sent as a broadcast message on the current channel 130b using MAC.I 106n. In exemplary embodiments, for a step 205 each DPP message 122 for each key Br 102a is sent an exemplary 3 to 5 times. Note that for a step 205, multiple different DPP messages 122 for different keys Br 102a can be sent concurrently in order to speed the process for responder 102' operating with responder DPP configuration 131 and a compatible network 107 for key Br 102a to
 25 receive the message 122 via WiFi link 192.

At step 206, initiator 101 or device 106 can determine if a DPP authentication response message 123 for a transmitted DPP authentication request message 122 has been received. If a DPP authentication response message 123 has been received, then initiator 101 and device 106 can conclude
 30 the step 200 in order to establish communications with a responder 102 or responder proxy 102' (by receiving a DPP authentication response message 123). If a DPP authentication response message 123 has not been received, then initiator 101 and device 106 can proceed to step 207.

At step 207, initiator 101 or device 106 can determine if device 106 has cycled through all available channels 130b on all available operating classes 130a for WiFi radio 106r operating in device
 35 106. If device 106 has not cycled through all available channels 130b on all available operating classes 130a, then device 106 can (i) increment a channel number 130b from the recorded DPP configuration

130 or (ii) if device 106 is already operating at the highest channel number 130b in an operating class 130a then change the operating class 130a for radio 106r and implement the lowest channel number in the new operating class 130a, and then return to step 205 to retransmit the collection of DPP messages 122 for each key Br 102a on the new channel number.

5 At step 207, if device 106 has cycled through all available channels 130b on all available operating classes 130a, then device 106 can (i) store data from a step 204 in nonvolatile memory for access at a later step 200, and (ii) conclude step 200 and enter a period of sleep or dormancy. After a period of time, device 106 could then wake and repeat the series of steps in a step 200 in order to receive a DPP authentication response 123 message from a responder proxy 102'. Note that upon
10 conducting a step 200 again at a later time, for a step 204 device 106 or initiator 101 could read (a) the data previously calculated or derived from a prior step 204 in (b) nonvolatile memory to access data for sending DPP messages 122 again, instead of re-calculating all the steps in 204.

Note that a step 200 can also be conducted for each set of parameters 199 supported by device 106, such as using a first step 200 with a first bootstrap private key bi 101b with a first set of parameters
15 199, and then a second step 200 concurrently with a second bootstrap private key bi 101b with a second set of parameters 199. Using the step 200 depicted for a device 106 operating an initiator 101 can allow device 106 to automatically receive a DPP authentication response message 123 from a responder proxy 102' when device 106 is powered on and with range of a responder proxy 102' for a responder bootstrap public key Br 102a (where responder 102' communicates with a DPP server 103
20 that securely records the responder bootstrap private key br 102b). A step 200 could be conducted with a responder 102* from Figure 1b or Figure 1c as well, such that the use of a responder proxy 102' is not required to use a step 200.

Figure 3a

25 Figure 3a is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device database, a DPP server, a responder proxy, and an initiator, in accordance with exemplary embodiments. System 300a can include a device database 104, DPP server 103, responder proxy 102', and initiator 101 operating in device 106. Responder proxy 102' can communicate with DPP server 103 via IP network 113 as depicted in Figure 1a. As
30 contemplated herein, a responder proxy 102' depicted in Figure 3a and subsequent Figures below may also be referred to herein as responder 102', and a responder 102' and responder proxy 102' may be equivalent. In other words, a responder 102' may operate as a responder with some functions and operations for DPP 191 by proxy with DPP server 103.

35 Before starting steps and message flows depicted in Figure 3a, the device 106 with initiator 101 can record responder bootstrap public keys Br 102a depicted for an initiator key database 101t

from Figure 1d, where initiator 101 records at least one key Br 102a. The key Br 102a for a initiator 101 could be received in an “out of band” message 121 from Figure 1b and Figure 1c, or the key Br 102a could be recorded in a device 106 during manufacturing or distribution. Initiator key database 101t for device 106 could also record a plurality of responder bootstrap public keys Br 102a in a table 101ta from Figure 1d. For embodiments supporting mutual authentication, then device 106 can also record at least one initiator bootstrap public key Bi 101a and a corresponding initiator bootstrap private key bi 101b, such as in a table 101tb from Figure 1d. A device 106 with initiator 101 can also operate with at least one device supported configuration 130z from a DPP configuration 130 in Figure 1d. Exemplary PKI keys for initiator 101 in device 106 could also comprise the exemplary keys depicted in Figure 1e. Device database 104 can record data depicted and described in connection with Figure 2a.

Responder 102' can be different than responder 102* in Figure 1b and Figure 1c, since (a) some PKI and cryptographic computations (plus recording of PKI keys) that would normally be performed by a responder 102* as contemplated by DPP specification version 1.0 can (b) be performed by DPP server 103 and thus responder 102' is different than an responder 102*. For system 300a, DPP server 103 and device database 104 may establish a secure session setup 302a, which could comprise establishing a secure communications link between the two servers using protocols such as TLS, IPSec, a virtual private network (VPN), a secure shell (SSH), or similar networking, transport, or application layer technologies in order to establish secure communications between DPP server 103 and database 104. Secure session setup 302a can utilize a certificate cert.DPPS 103c for DPP server 103 and a certificate for database 104 in order to provide mutual authentication and mutual key derivation for a symmetric encryption key in secure session 302a.

At step 303, responder 102 can be powered on and begin operating a WiFi radio 102x with a responder configuration 131 as depicted in Figure 1f. At step 303, responder 102 can operate as an unconfigured responder 102 in Figure 1f, where the subsequent responder configuration step 127 depicted in Figure 3a can convert responder 102 into a configured responder 102'. In other words, responder 102' can comprise a responder 102 that has undergone a configuration step 127. At step 303, responder 102 can use a first MAC address MAC.R 102n to transmit or send and receive wireless data via WiFi link 192. At step 303, responder 102 can use a second, network MAC address MAC.N 102q in order to send and receive data with an IP network 113 and network 107 including DPP server 103. Or, in some exemplary embodiments, the same physical interface and MAC address could be used for both MAC.R 102n and MAC.N 102q.

DPP server 103 and responder 102 can establish a secure session 302b, where secure session 302b could comprise a session using TLS, DTLS, IPSec, a VPN, and other possibilities exist as well. Responder 102 can previously record a URL or domain name to reach DPP server 103. Responder 102 can also use a certificate for DPP server 103 comprising cert.DPPS 103c recorded in responder

102 in order to locate and connect with DPP server 103. As depicted in Figure 3a, secure session 302b could also comprise DPP server 103 receiving and using a responder certificate cert.responder 102m in order to authenticate responder 102. Alternatively, DPP server 103 could authenticate a user of responder 102 via a user ID and password entered by user of responder 102 into a user interface 102k for responder 102. In some exemplary embodiments, authentication of a responder 102 or a user of responder 102 for a secure session 302b could be omitted, such as if responder 102 does not record a cert.responder 102m. Secure session 302b could be established using cert.DPPS 103c, similar to a web site providing HTTPS security based on a certificate for the web site (i.e. the session 302b can authenticate the server and secure the session based on the certificate from the server). In exemplary
10 embodiments, responder 102 can authenticate the cert.DPPS 103c using parent certificates up to a recorded root certificate in cert.root 102w from Figure 1f.

After secure session 302b has been established between DPP server 103 and responder 102, DPP server 103 can conduct a step 304. Note that a responder identity 102i for responder 102 could be received by DPP server 103 during setup of secure session 302b. At step 304, DPP server 103 can
15 select a DPP application 102g for responder 102', where DPP application 102g can comprise software or firmware for an unconfigured responder 102 to load and operate in order to function as a configured responder 102'. DPP application 102g was depicted and described in Figure 1f. DPP server 103 could select DPP application 102g based on the OS 102e for responder 102, which could be communicated and received by DPP server 103 in setup of secure session 302b. DPP server 103 could obtain DPP
20 application 102g from another server in network 107 or external networks. At step 304, DPP server 103 could also select or process a responder network table 102t for responder 102, where responder network table 102t was depicted and described in connection with Figure 1f. Responder network table 102t can select which DPP server 103 a configured responder 102' should forward an incoming DPP authentication request message 122 based on the responder bootstrap public key hash value 102aa.
25 Although a DPP server 103 is depicted in Figure 3a as conducting secure session 302b and a step 304 and then a step 306, another server associated with network 107 could perform those functions and steps for a DPP server 103.

At step 306, DPP server 103 can select the responder mode 306a of operation for responder 102' to conduct a DPP 191 for receipt and processing of incoming DPP authentication request
30 messages 122 from an initiator 101. The selected mode of operation for responder 102' can comprise a mode 306a or a responder mode 306a, which could be recorded in a server database 103a as depicted and described in connection with Figure 3c below. The selected mode 306a in a step 306 can specify the distribution of PKI keys and related processing between DPP server 103 and responder 102'. The selected mode 306a in a step 306 can determine the subsequent series of steps and message flows
35 between DPP server 103 and responder 102'. The selected mode 306a in a step 306 could be based on the security established or mutual authentication or server authentication in secure session 302b.

As one example, if secure session 302b used a certificate cert.DPPs 103c but not a certificate cert.responder 102m (e.g. secure session 302b is less secure or not mutually authenticated or responder 102 is not securely authenticated), then selected mode 306a can comprise a first mode depicted in Figure 3a and Figure 3b, where DPP server 103 records and operates on responder ephemeral private key pr 102r and also all secret shared keys k1 314a, k2 314c, and ke 327a. As a second example, if secure session 302b is mutually authenticated (e.g. used a certificate cert.DPPs 103c and a certificate cert.responder 102m), then a selected mode 306a in a step 306 could comprise a second mode 306a for responder 102' to operate as depicted in Figure 5a, where responder 102' records and operates on responder ephemeral private key pr 102r (and DPP server 103 does not record or operate on the key pr 102r).

As another example, a selected mode 306a in a step 306 could also depend on an identity for device 106, such as MAC.I 106n. For embodiments where the selected mode 306a of operation of a responder 102' depends on the identity of a device 106, then a step 306 to select the mode could be conducted after receipt of a DPP message 122 which would include MAC.I 106n in order to identify the device 106. In other words, in some exemplary embodiments, a preferred responder mode 306a for a device 106 could depend on the device 106, such as if device 106 was a higher-value piece of equipment such as an automobile compared to a potentially lower value networked sensor. For these embodiments, the responder mode 306a for responder 102' could be specified in a device database 104, and for these embodiments the selected mode 306a for responder 102' to operate could be conducted by a step 306 after the receipt of message 312a with data from device database 104.

A first mode 306a could (a) specify that DPP server 103 records all public and private bootstrap and ephemerals keys for responder 102', plus the public keys Bi 101a and Pi 101e for initiator 101 such that (b) resulting cryptographic operations and ECDH key exchanges and symmetric ciphering reside within DPP server 103 and results are transmitted to responder 102'. This first mode 306a could represent the message flows and steps in Figure 3a and Figure 3b, with the recording of keys depicted in Figure 3c below. A second mode 306a could specify that DPP server records and operates on the responder bootstrap private key br 102b, but responder 102' records and operates on the responder ephemeral private key pr 102r, and the second mode 306a is depicted in Figure 5a below. A third responder mode 306a could specify that both responder 102' and DPP server 103 record and operate on responder ephemeral private key pr 102r, such that (b) resulting cryptographic operations and key exchanges are shared between DPP server 103 and responder 102'. This third responder mode 306a could represent the message flows and steps depicted in Figure 5b, with the recording of keys depicted in Figure 3c below.

In exemplary embodiments, all the first, second, and third responder modes 306a specify responder bootstrap private key br 102b residing within DPP server 103 and not shared with responder 102'. By not sharing responder bootstrap private key br 102b with responder 102' (which could be a

device with much less security than DPP server 103), the responder bootstrap private key br 102b can remain secured within network 107. These embodiments where responder bootstrap private key br 102b remains in network 107 (e.g. in DPP server 103 and device database 104) support securely sharing the same responder bootstrap public key Br 102a potentially across a plurality of devices 106 with initiators 101 (as well as responder 102' recording key Br 102a). Sharing the same responder bootstrap public key Br 102a across multiple devices supports recording the key Br 102a in a device 106 either (i) during manufacturing or distribution of device 106, or (ii) at a time before any association between device 106 and network access point 105 is known. This simplifies and enables the use of a device provisioning protocol 191 across multiple devices 106 with mutual authentication, where the responder bootstrap public key Br 102a can comprise a shared key 102z. Or, in other embodiments the responder bootstrap public key Br 102a may not be a shared key 102z, but rather unique for a responder 102' (such as a unique responder bootstrap public key Br 102a recorded in a tag 102v), and security is still enhanced because responder 102' may not record or operate on responder bootstrap private key br 102b.

In other words, without the technology described in the present disclosure, a device provisioning protocol with mutual authentication can essentially shift the problem of (a) securely mutually authenticating a device and providing credentials 109 to a device 106 to a new, but related problem of (b) securely providing a responder bootstrap private key br 102b to an responder 102*, where an initiator should record the responder bootstrap public key Br 102a. The first, second, and third modes 306a can solve this shifted/new problem since (i) the responder bootstrap private key 102b can permanently reside in a network 107 and does not need to be sent to the responder 102', and (ii) an initiator can be configured to record key Br 102a before receipt of device 106 by a user (since the same key Br 102a can be written to many different devices). However, with the technology in the present disclosure, the responder 102' can still transmit and receive the full set of a device provisioning protocol 191 messages with an initiator 101 as specified in DPP specification version 1.0.

A fourth mode 306a could specify that responder 102' records all of the responder bootstrap and ephemeral PKI keys, as well as the initiator 101 public keys, such that resulting cryptographic operations and key exchanges reside in responder 102'. This fourth mode 306a could represent the message flows and steps depicted in Figure 1b or Figure 1c, and in this case responder 102' could comprise an initiator 102*. Other possibilities exist as well without departing from the scope of the present disclosure. Additional details for the different responder modes 306a selected in a step 306, with the resulting different allocation or recording of PKI keys and derived symmetric ciphering keys are described below in Figure 3b for a server database 103a.

After a step 306, DPP server 103 can send responder 102 a message 305, where message 305 can include a responder identity 102i, responder network table 102t, responder DPP application 102g, and a responder configuration 131. Note that the responder configuration 131 in message 305 can

represent an updated responder configuration to replace or update a previous responder configuration 131 recorded by responder 102. DPP application 102g could comprise a link or URL for responder 102 to download and install DPP application 102g (comprising such as less than an exemplary 100 bytes), or the full software DPP application 102g for responder 102' to operate, which could comprise
5 more than 1 megabyte of data. Responder identity 102i can be useful for a device such as access point 105 operating a responder 102', since in exemplary embodiments an access point 105 could operate more than one responder 102', and the different responders 102' operating in an access point 105 or other device could be identified using the responder identity 102i. In some exemplary embodiments, the responder mode 306a could also be sent in a message 305 from DPP server 103. In other exemplary
10 embodiments, responder mode 306a for responder 102' can also be sent below in a message 312b, after an identity for device 106 such as MAC.I 106n is obtained from a subsequent message 122.

Responder 102 using responder identity 102i in system 300a can receive message 305 and process the data. Responder 102 can download and install the DPP application 102g from a message 305. Responder 102 can load and apply both responder network table 102t and the received responder
15 configuration 131, which could specify radio frequencies or channels to utilize, such as class 131a and channels 131b. Responder 102 can conduct the responder configuration step 127 depicted and described in connection with Figure 1f in order to begin operating as a configured responder 102'. At step 307, responder 102' can begin listening for incoming DPP authentication request messages 122 using WiFi radio 102x and operating with responder configuration 131 in order to communicate with
20 a WiFi radio 106r operating in device 106 with initiator 101, where the communications can use a WiFi data link 192. In exemplary embodiments, DPP application 102g may be (i) included or distributed along with an operating system 102e running on responder 102' or (ii) acquired at a different time than a message 305, and thus from a separate download of DPP application 102g. In exemplary embodiments, WiFi radio 102x operating in responder 102' can listen for message 122
25 either broadcast or unicast messages or datagrams from device 106 with initiator 101 when using responder configuration 131. In exemplary embodiments, initiator 101 transmits DPP messages 122 using the same or compatible configuration as responder configuration 131.

Device 106 operating initiator 101 can conduct the steps 200 depicted and described in connection with Figure 2b above. In summary, at step 200 in Figure 3a, device 106 may power on
30 from an unpowered or "deep sleep" state. Power could be provided from a battery or converted "wall power". Device 106 could (i) load the equivalent or initiator version of information contained in responder configuration 131 using a bootloader program and (ii) power on WiFi radio 106r. Device 106 can load database 101t for initiator 101 with at least one responder bootstrap public key Br 102a. As a portion of a step 200, device 106 can conduct a step 204 as depicted in Figure 2b. Step 204 can
35 comprise steps 313, 314, 315, and 316. Note that step 200 can comprise a cycle or loop through responder bootstrap public keys Br 102a recorded in database 101t so that initiator 101 send a DPP

authentication request message 122 for each responder bootstrap public key Br 102a until initiator 101 successfully receives a DPP authentication response message 123.

A step 313 can comprise initiator 101 deriving an initiator ephemeral private key pi 101d and an initiator ephemeral public key Pi 101c. Step 313 can use a random number generator to produce a random number, which could be input into a key pair generation algorithm using the set of cryptographic parameters 199a. The resulting PKI key values from conducting a step 313 can comprise the initiator ephemeral private key pi 101d and the initiator ephemeral public key Pi 101c. The data for a step 313 comprising derived ephemeral PKI keys for initiator 101 can be recorded in memory of device 106, such that the ephemeral PKI keys can be derived one time until a DPP authentication response message 123 is received from a responder such as responder 102'.

Step 314 can comprise initiator 101 conducting a first initiator key exchange step 314 using an Elliptic Curve Diffie Hellman (ECDH) key exchange 401a, as depicted and described in connection with Figure 4a below. Data input for a step 314 can comprise the initiator ephemeral private key pi 101d and the selected responder bootstrap public key Br 102a from a step 203. Data output from a step 314 can comprise a key k1 314a as described in Figure 4a below. Step 314 can comprise the key exchange and symmetric key derivation step in DPP specification version 1.0 section 6.2.2 in order for an initiator to obtain key k1 314a. Note that step 314 by initiator 101 can be conducted for each recorded responder bootstrap public key Br 102 in a database 101t. The data for a step 314 can be stored in memory for device 106 such that keys k1 314a for each key Br 102a can be stored and reused for a subsequent "loop" in a step 200 until initiator 101 receives a DPP authentication response message 123, thereby reducing the number of computations and energy required by device 106.

At step 315, initiator 101 can generate a random number for an initiator nonce 317a using parameters 199a. As used herein and throughout the present disclosure, the words "random number" can also mean a pseudo random number, since random numbers generated by nodes may not be truly random. However, their "randomness" may be sufficient in order to securely conduct the cryptographic operations contemplated herein. Initiator 101 can also determine the initiator capabilities, which could comprise an enrollee for the initiator as depicted in Figure 3a. In other embodiments, the responder 102' could have initiator capabilities as an enrollee (where the initiator 101 could be the configurator). A step 315 can comprise encrypting the initiator nonce 317a and initiator capabilities using the derived key k1 from step 314. An encryption step 315 step is further depicted and described in connection with Figure 4a below.

Note that step 315 by initiator 101 can be conducted for (i) each recorded responder bootstrap public key Br 102a in a database 101t with (ii) a corresponding key k1 314a recorded from a step 314. A single initiator nonce 317a can be reused across all encryption steps 315 and the output of all encryption steps 315 comprising ciphertext 315a can be recorded in memory of device 106, such that an encryption step 315 needs only to be conducted once by initiator 101 for each key Br 102a in a

database 101t (by reusing initiator ephemeral private key pi 101d and key $k1$ 314a). In other words, initiator 101 can record one ciphertext 315a for each responder bootstrap public key Br 102a in a database 101t, and a subsequent “loop” or generation of a DPP authentication request message 122 (such as transmitting on a different channel 130b) can reuse the ciphertext 315a recorded for the key Br 102a, key pi 101d, and initiator nonce 317a. In this manner, the number of computations and energy use by device 106 can be reduced, by reusing a previously recorded ciphertext 315a when re-transmitting a DPP authentication request message 122 for the same key Br 102a at a later time.

At step 316, initiator 101 can calculate secure hash value 101aa for initiator bootstrap public key Bi 101a and secure hash value 102aa for responder bootstrap public key Br 102a, using a secure hash algorithm 199e. The values are depicted in Figure 3a as “H(Bi)” and “H(Br)”, respectively. Initiator 101 can then transmit using radio 106r and channel 130b a DPP authentication request message 122, which can include the hash value 101aa for the initiator bootstrap public key Bi 101a and the hash value 102aa the responder bootstrap public key Br 102a, the derived initiator ephemeral public key Pi 101c, and ciphertext 315a, where ciphertext 315a is encrypted with key $k1$ 314a and includes an initiator nonce 317a and the initiator capabilities.

In exemplary embodiments and as depicted and described in connection with Figure 2b in a step 205 for a step 200 depicted in Figure 3a, a DPP authentication request message 122 for each recorded responder bootstrap public key Br 102a can be transmitted by initiator 101 on channel 130b. In exemplary embodiments, each DPP message 122 can be transmitted as a broadcast message on channel 130b, since initiator 101 may not know a responder 102’ MAC address. As depicted for a step 200 above in Figure 2b, initiator 101 may need to repeat messages 122 for each key Br 102a multiple times on several different channels in a channel list 130b, since (i) responder 102’ may be listening on a different channel in a responder channel list 131b, and (ii) responder 102’ may be using only one of the bootstrap responder public keys Br 102a from a database 101t which corresponds to the key Br 102a recorded and operated by responder 102’.

Although a message 122, message 123, etc. as depicted herein shows key values of “ $k1$ ”, “ ke ”, etc., the depiction of the keys is to illustrate that ciphertext is encrypted with the corresponding key by the use of brackets. In other words, although the key values are shown with the messages, the actual key values are not transmitted inside or with the message in exemplary embodiments. The equivalent syntax is used in DPP specification version 1.0.

Responder 102’ can receive a DPP authentication request message 122 in a step 308a and process the message. For a step 308a, responder 102’ can compare (i) hash values 102aa for H(Br 102a) received in message 122 with (ii) the recorded hash values 102aa in a responder network table 102t. Or, responder 102’ can record the responder bootstrap public key Br 102a and calculate a hash value and compare with the hash value received in a message 122. If (a) the hash value 102aa for H(Br 102a) in message 122 matches or are equal to a recorded or calculated hash value 102aa, then

(b) for a step 308b responder 102' can conduct a key validation step on received initiator ephemeral public key P_i 101c described two paragraphs below. Otherwise, responder 102' can continue to monitor or listen for new, different incoming messages 122 where the hash value 102aa would equal a recorded or calculated hash value 102aa for a responder bootstrap public key Br 102a in responder 102.

An identity for device 106 and initiator 101 could also comprise the secure hash value over the initiator bootstrap public key Bi 101a, which can be value 101aa as depicted for message 122 in Figure 3a. In other words, as contemplated throughout the present disclosure, an identity for device 106 and/or initiator 101 can comprise the hash value 101aa over the initiator bootstrap public key Bi 101a. For example, a device database 104 (Fig 2a), a server database 103a (Fig 3c), or responder 102' could record and use hash value 101aa as an identity for device 106 or initiator 101 in order to select keys for conducting key exchange and encryption/decryption steps. The hash value 101aa could be either (i) instead of or (ii) in addition to $MAC.I$ 106n or device identity $ID.device$ 106i in order to identify and track device 106 with initiator 101 across a potential plurality of devices 106 in systems 100, 300, 500, etc.

A step 308b could comprise responder 102' performing the steps for an ECC Full Public-Key Validation Routine in section 5.6.2.3.2 of FIPS publication SP 800-56A (revision 2) for the received initiator ephemeral public key P_i 101c. Alternatively, step 308b can comprise responder 102' performing the steps ECC Partial Public-Key Validation Routine in section 5.6.2.3.3 of the same FIPS publication. Other example steps within a public key validation step 308 can comprise (i) verifying the public key is not at the "point of infinity", and (ii) verifying the coordinates of the point for the public key are in the range $[0, p-1]$, where p is the prime defining the finite field. Other possibilities exist as well for evaluating and validating a received public key is cryptographically secure in a public key validation step 308b, without departing from the scope of the present disclosure. As contemplated in the present disclosure, a responder 102', a DPP server 103, or an initiator 101 can conduct a public key validation step 308b each time a public key is received. In addition, a responder network table 102t can include a set of parameters 199a for each entry corresponding to a key Br 102a, and the set of parameters 199a could be used for key validation steps in a step 308b.

A step 308c can comprise responder 102' selecting a DPP server 103 based on the received hash value 102aa for responder bootstrap public key Br 102a in message 122. A responder 102' could communicate with a plurality of different DPP servers 103 or networks 107, and a key Br 102a could be associated with different DPP servers 103. For example, different networks 107 or DPP servers 103 could record and operate with the responder bootstrap private key br 102b in order to conduct or support a DPP 191 with initiator 101. The responder 102' could use the hash value 102aa and a responder network table 102t as depicted in Figure 1f in order to determine which DPP server 103 should receive data from the incoming DPP authentication request message 122. Figure 3a depicts

hash value 102aa for DPP server 103 in network 107 from Figure 1a, although hash value 102aa could be for a different network 107 in exemplary embodiments. In other words, DPP server 103 could comprise a depicted “DPP 103 -1” for responder network table 102t in Figure 1f, and a different hash value 102aa for Br 102a in message 122 could be for a different DPP server 103 comprising a depicted value of “DPP 103 -3” in responder network table 102t in Figure 1f. A step 308c can also include responder 102’ starting a timer in order to count the seconds between receiving message 122 and sending upcoming message 123. Additional details regarding the timer from a step 308c are described in a step 324 below.

For some exemplary embodiments, a responder network table 102t as depicted in Figure 1f above could (i) omit the storing or recording of a plurality of responder bootstrap public key hash values 102aa and (ii) include at least one value or identity or domain name for a DPP server 103. Thus, although “DPP server 103” is depicted for a responder network table 102t in Figure 1f, the value recorded could be for an identity or domain name for a DPP server 103, such that responder 102’ could forward data received in a message 122 to the DPP server 103 in a subsequent message 309. For a step 308c, a responder 102’ could select the DPP server 103 and an identity or domain name for DPP server 103 from the responder network table 102t for sending a subsequent message 309 with data received in DPP authentication request 122. For these embodiments where responder network table 102t omits the storing or recording of a plurality of responder bootstrap public key hash values 102aa, then a DPP server 103 could record a table that stores a plurality of responder bootstrap public key hash values 102aa associated with a plurality of DPP servers 103. DPP server 103 could take actions after receiving a message 309 in order to select another DPP server 103 associated with the responder bootstrap public key hash values 102aa received in a message 309 (for embodiments where the received responder bootstrap public key hash value 102aa from a message 309 does not match the DPP server 103 that receives the message 309).

After steps 308a, 308b, and 308c, responder 102’ can send DPP server 103 a message 309 via the secure session 302b. Message 309 can include a responder identity 102i, an identity for initiator 101 and/or device 101 comprising MAC.I 106n, and the data contained in DPP authentication request message 122 received by responder 102’. Note that MAC.I 106n could be observed by responder 102’ as the source MAC address for message 122 received via WiFi link 192. Data from message 122 in message 309 can include secure hash value 101aa for initiator bootstrap public key Bi 101a and secure hash value 102aa for responder bootstrap public key Br 102a, the initiator ephemeral public key Pi 101c, and ciphertext 315a. Although not depicted in Figure 3a, a message 309 could also include the responder bootstrap public key Br 102a, for embodiments where responder 102’ also records key Br 102a. DPP server 103 can receive message 309 and process the data, including recording the received initiator ephemeral public key Pi 101c in a server database 103a, which is also depicted in Figure 3c

below. Before taking steps to process the received ciphertext 315a, DPP server 103 could require additional PKI keys, if they are not already recorded in a server database 103a.

For some embodiments, DPP server 103 can store the equivalent of a responder network table 102t depicted in Figure 1f (and responder 102' may not record the plurality of responder bootstrap public key hash values 102aa for different DPP servers 103). For these embodiments, after receipt of a message 309, DPP server 103 could use the received value of responder bootstrap public key hash value 102aa from a message 309 to query or obtain a different, second DPP server 103 associated with the responder bootstrap public key hash value 102aa. DPP server could forward the data from the message 309 to the different, second DPP server 103. Note that the different, second DPP server 103 could be associated with a different network 107. The different, second DPP server 103 could received the message 309, and subsequently the different, second DPP server 103 could conduct the subsequent steps for a DPP server 103 depicted in Figure 3a and Figure 3b, such as sending a message 312b, conducting the depicted steps of 318 through 315b, sending a message 313 to responder 102', etc.

For embodiments where DPP server 103 does not already record PKI keys for responder 102' and initiator 101, DPP server 103 can send device database 104 a message 310 in order to query for keys and information in order to support a DPP 191 between responder 102' and initiator 101. DPP server 103 can send message 310 through secure session 302a. Message 310 can include an identity for device 101, which could be MAC.I 106n and secure hash value 101aa for initiator bootstrap public key Bi 101a and secure hash value 102aa for responder bootstrap public key Br 102a.

Device database 104 can receive message 310. Device database 104 in step 312 can conduct a query of device database 104 for data associated with the received data from an initiator comprising and identity of MAC.I 106n, hash value 102aa and hash value 101aa. Exemplary data for device database 104 was depicted and described in connection with Figure 2a above. Not all data in the depicted message 310 is required to conduct a query in a step 312, and a hash value 101aa for Bi 101a could be omitted as one example. Or, in exemplary embodiments, the MAC.I 106n may not be recorded in a device database 104, but data for a device 106 could be found using hash value 101aa for Bi 101a. Thus, in exemplary embodiments at least one of MAC.I 106n and hash value 101aa for Bi 101a is received by device database 104 in a message 310, and the at least one MAC.I 106n and hash value 101aa for Bi 101a is recorded in a device database 104.

In exemplary embodiments, a query in a step 312 can select a responder bootstrap private key br 102b, responder bootstrap public key Br 102a, optionally either (i) initiator bootstrap public key Bi 101a or (ii) PKEX key 198 for device 106 from a device database 104. For embodiments where initiator bootstrap PKI keys for device 106 are updated over time, then the most recently updated initiator bootstrap public key Bi 101a can be selected by a database query in a step 312. Additional data can be selected for device 106 and/or initiator 101 in a query at step 312 as well, including a set of parameters 199a to use with bootstrap PKI keys and ephemeral keys and/or a responder mode 306a

for DPP server 103 and responder 102' to use when conducting a DPP 191 with initiator 101. Note that a responder mode 306a could be selected for responder 102' in either a step 306 or a query in step 312. The query in step 312 can return values for a message 312a from device database 104 to DPP server 103.

5 The values in a message 312a can comprise data for a device identity of MAC.I 106n (or hash value 101aa) and the values can include responder bootstrap public key Br 102a, initiator bootstrap public key Bi 101a, responder bootstrap private key br 102b, a device identity ID.device 101i, a selected set of cryptographic parameters 199a for the PKI keys, and optionally PKEX key 198. MAC.I 106n or hash value 101aa can be included in a message 312a so that DPP server 103 can track which
10 of a possible plurality of initiators 101 and devices 106 that message 312a is associated with. Although not depicted for a message 312a, a message 312a could also include a responder mode 306a for device 106 using MAC.I 106n.

DPP server 103 can receive the message 312a and record the data received in a server database 103a. DPP server 103 can (i) use a device identity such as MAC.I 106n or hash value 101aa in order
15 to select or update a responder mode 306a previously selected in a step 306, or (ii) conduct a step 306 at this time in order to select a responder mode 306a recorded for responder 102' in a server database 103a, where a server database 103a is depicted and described in connection with Figure 3c below. As depicted in Figure 3a, DPP server 103 can then send message 312b to responder 102', where message 312b can include (i) the selected responder mode 306a of operation for responder 102' and DPP server
20 103 to use when conducting a DPP 191 with initiator 101, (ii) a selected set of cryptographic parameters 199a for use with PKI keys and a device provisioning protocol, and (iii) optionally a PKEX key 198 for conducting a public key exchange with initiator 101. In exemplary embodiments, the PKEX key 198 can be included in message 312a for embodiments where device database 104 does not record an initiator bootstrap public key Bi 101a for device 106 with MAC.I 106n. For the
25 responder mode 306a depicted in Figure 3a and Figure 3b, message 312b can comprise a signal that DPP server 103 derives the responder ephemeral private key 102r. Message 312b can also optionally include a device identity such as MAC.I 106n or hash value 101aa in order to associate the data in message 312b with a particular initiator 101, since responder 102' could potentially communicate with a plurality of different devices 106. For some exemplary embodiments, DPP server 103 can send
30 message 312b after establishment of secure session 302b and before DPP server 103 receives a message 309.

For some embodiments, DPP server 103 can (i) store the equivalent of a responder network table 102f depicted and described in connection with Figure 1f, (ii) may not store or operate with the responder bootstrap private key br 102b for responder 102', and (iii) select a different, second DPP
35 server 103 after the receipt of a message 309 as described six paragraphs above. For these embodiments, a message 312b can (i) include an identity or domain name for the different, second

DPP server 103 and (ii) omit depicted data of responder mode 306a and parameters 199 and PKEX key 198. For these embodiments, responder 102' (i) could receive the message 312b and use the received identity or domain name of the different, second DPP server 103 in the message 312b to (ii) resend the message 309 to the different, second DPP server 103 received in the message 312b. Responder 102' and the different, second DPP server 103 could establish a second secure session 302b. For these embodiments, after resending the message 309 to the different, second DPP server 103, the subsequent steps and messages depicted in Figures 3a and 3b can apply for the different, second DPP server 103 with the responder 102' (where the different, second DPP server 103 operates as the depicted DPP server 103 in Figure 3a and Figure 3b). For example, the different, second DPP server 103 could send a message 312b to responder 102', conduct the depicted steps of 318 through 315b, send the message 313 to responder 102', etc.

Responder 102' can receive message 312b from DPP server 103 and conduct a step 312c. In step 312c, responder 102' can use and record the responder mode 306a in order to conduct subsequent steps for responder 102' and messages with DPP server 103 for device 106 with initiator 101. Note that a responder mode 306a could specify the operation of responder 102' with device 106, such that responder 102' could operate with a first responder mode 306a for a first device 106 and a second responder mode 306a with a second device 106. The depicted subsequent steps and messages for responder 102' in Figure 3a with the responder mode 306a depicted as received in a message 312b can comprise a first responder mode 306a depicted as "Fig. 3" in server database 103a. For embodiments where a responder network table 102t includes cryptographic parameters 199a associated with the responder bootstrap public key hash value 102aa, then a message 312b could optionally omit parameters 199a within message 312b, and responder 102' can determine the cryptographic parameters 199a based on the hash value 102aa received from initiator 101 in a message 122.

For embodiments where PKEX key 198 is received in a message 312b (e.g. embodiments where device database 104 may not previously record initiator bootstrap public key Bi 101a before step 312), then responder 102' can conduct a public key exchange protocol exchange 312d with initiator 101 using the PKEX key 198 depicted in Figure 3a. A PKEX 312d can comprise the PKEX protocol as specified in section 5.6 of DPP specification version 1.0. In this manner, responder 102' can receive the key Bi 101a if not previously recorded by a device database 104. Although not depicted in Figure 3a, upon conclusion of a PKEX 312d (or otherwise obtaining an initiator bootstrap public key Bi 101a such as an "out of band" message 126 in Figure 1c), responder 102' can forward the key Bi 101a to DPP server 103 after receipt of the initiator bootstrap public key Bi 101a from a step 312d. The use of a PKEX protocol and key 198 can also be optionally omitted in some embodiments, if initiator bootstrap public key Bi 101a can be recorded by a network 107 prior to a step 312c.

As depicted in Figure 3a, the collection of steps and messages in Figure 3a, from step 200 for device 106 through the optional message 312d can be grouped together into a step 301. Step 301 with the collective group of sub-steps and messages depicted for a step 301 can also be performed in other embodiments depicted in Figure 5a and Figure 5b below. Subsequent messages and steps depicted in Figure 3a starting after step 301 can represent a first responder mode 306a for DPP server 103 and responder 102', where the allocation of recording PKI keys and symmetric keys from key exchanges for the first mode 306a are depicted below in Figure 3c. The first responder mode 306a can be depicted in Figure 3c with the label "Fig. 3" is depicted in Figure 3a and Figure 3b.

For the first responder mode 306a selected in a step 306, DPP server 103 can then conduct the series of depicted steps 318 through 315b in Figure 3a in order to generate data for responder 102' to send a DPP authentication response message 123. At step 318, DPP server 103 can also generate two random numbers using a random number generator. The first random number can be used as a responder ephemeral private key pr 102d. The responder ephemeral private key pr 102p can be input into an ECC key pair generation algorithm in order to output a responder ephemeral public key Pr 102p. The set of cryptographic parameters 199a used for the responder ephemeral PKI key pair can match the recorded parameters 199a used with the bootstrap public keys such as responder bootstrap public key Br 102a. The second random number generated by DPP server 103 can be used as a responder nonce 318b. DPP server 103 can also determine responder capabilities 319b, which could comprise the capabilities for a configurator. The responder ephemeral private key pr 102d can also be referred to as responder protocol private key pr 102d, and responder ephemeral public key Pr 102c can be referred to as responder protocol public key Pr 102c. The second random number in a step 318 can comprise a responder nonce value 318b, which can be used in subsequent messages and calculation of authentication values.

DPP server 103 can conduct a first responder key exchange step 319, where the details for a key exchange step 319 are depicted in Figure 4a below. In summary, the received initiator ephemeral public key Pi 101c and the responder bootstrap private key br 102b can be input into an ECDH key exchange in order to mutually derive a shared secret key M 410. The shared secret key M 410 can be input into a key derivation function 402a in order for DPP server 103 to derive mutually shared key k1 314a. A DPP server 103 can then conduct a decryption step 320 as depicted in Figure 4a below. DPP server 103 can input the received ciphertext 315a from a message 122 into a symmetric ciphering algorithm 405b and key k1 314a using parameter 199f in order to read the plaintext 403a, which could comprise the initiator nonce 317a and the initiator capabilities. In exemplary embodiments, the responder 102' can have capabilities of a configurator, and DPP server 103 can select initiator capabilities or an enrollee in a step 320. Steps 319 and steps 320 can also comprise the first four paragraphs of section 6.2.3 of DPP specification version 1.0.

DPP server 103 can then conduct a second responder key exchange step 314b, where the details for a key exchange step 314b are depicted and described in connection with Figure 4b below. In summary, the initiator ephemeral public key P_i 101c received in message 122 can be input into an ECDH key exchange algorithm 401a along with the responder ephemeral private key pr 102r derived in a step 318 above in order to output a secret shared key N 411. The value for shared secret key N 411 can be recorded and used in the derivation of a key ke 327a in a third key exchange step 321 below. For a step 314b, the value for the shared secret key N 411 can be input into a key derivation function 402b in order to derive a mutually shared symmetric key k_2 314c.

DPP server 103 can conduct a third responder key exchange step 321, where the details for a key exchange step 321 are depicted in Figure 4c below. In summary, both (i) the derived responder ephemeral private key pr 102r from a step 318 with the recorded responder bootstrap private key br 102b, and (ii) the initiator bootstrap public key B_i 101a can be input into an ECDH key exchange algorithm 401c in order to output a secret shared key L 412. For embodiments where a DPP responder authentication 141 is conducted by DPP server 103 without the use of an initiator bootstrap public key B_i 101a, then calculation of a secret shared key L 412 can be optionally omitted in a step 321. Step 321 for DPP server 103 can then comprise a key derivation function 402c using the mutually derived secret shared keys M 410, N 411, and optionally L 412 in order to mutually derive secret shared key ke 327a. An example for a key exchange step 321 is described in point 2 of section 6.2.3 of DPP specification version 1.0 on page 48.

DPP server 103 can then conduct an encryption step 322, where the details for an encryption step 322 are depicted and described in connection with Figure 4c below. Step 322 can also comprise DPP server 103 creating a value R -auth 404, where R -auth 404 can comprise a hash of the initiator nonce 317a from plaintext 403a in step 320, the second random number R -nonce 318b from a step 318 above, and public PKI keys. The value for R -auth 404 in a step 322 can comprise the value in point 3 of section 6.2.3 of DPP specification version 1.0. An encryption step 322 can comprise DPP server 103 (a) encrypting the value for R -auth 404 using the mutually derived key ke 327a from the third key exchange step 321 above, where (b) the output of encryption step 322 can comprise ciphertext 324a.

DPP server 103 can then conduct an encryption step 315b, where the details for an encryption step 315b are depicted and described in connection with Figure 4b below. Ciphertext 324a from an encryption step 322 above can also be encrypted again for an encryption step 315b. The symmetric ciphering key for encryption step 315b can comprise symmetric key k_2 314c. Encryption step 315b, encryption step 322, and decryption step 320 can use a common symmetric ciphering algorithm 405a, which could comprise the Advanced Encryption Standard with Synthetic Initialization Vectors (AES-SIV) (and deciphering algorithm) also with a common set of symmetric ciphering parameters 199f. Symmetric ciphering parameters 199f can also specify the use of a block chaining mode such as cipher block chaining (CBC), counter mode (CTR), or Galois/Counter mode (GCM) and other possibilities

exist as well. In addition, symmetric ciphering parameters 199f could specify a mode for message authentication, which could comprise a CMAC mode as specified in NIST publication SP-800-38B. In exemplary embodiments, a symmetric ciphering algorithm 405a can comprise the AES-SIV algorithm as specified in IETF RFC 5297. The output from an encryption step 315b can be ciphertext 315c, as depicted in Figure 4b below.

After the encryption step 315b, DPP server 103 can send responder 102' a message 313, where message 313 can include an identity for device 106 such as MAC.I 106n, hash value 102aa for key Br 102a, hash value 101aa for key Bi 101a, the responder ephemeral (or protocol) public key Pr 102p from a step 318 and a ciphertext 315c. The ciphertext 315c can comprise the encrypted data depicted and described in connection with Figure 4b below from the encryption step 315b. As depicted in Figure 3a, ciphertext 315c can be encrypted with key k2 314c and include the responder nonce 318c, the initiator nonce 317a, the responder capabilities (as an exemplary configurator in Figure 3a), and a ciphertext 324a. In other words, ciphertext 324a can comprise data that is encrypted twice, where the first encryption with a key ke 327a can be in a step 322 as depicted in Figure 4c below, and the second encryption step can be with key k2 314c. Ciphertext 324a can include a responder authentication value 404, where the responder authentication value 404 is encrypted by DPP server 103 with key ke 327a.

Responder 102' can receive message 313 and conduct a step 324 to process the message. Note that message 313 and other messages between responder 102' and DPP server 103 can be sent and received via secure session 302b. Responder 102' can conduct a step 324. For a step 324, responder 102' can record the received responder ephemeral public key Pr 102p in a responder network table 102t along with the identity of device 106 comprising potentially MAC.I 106n. In other words, a responder network table 102t can be updated to include information received from network 107 (and DPP server 103) regarding devices 106 communication through network 107 via responder 102'. Step 324 can also include responder 102' conducting a key validation step 308b on the received key Pr 102p from message 313 using parameters 199a from a message 312b above. In exemplary embodiments, if message 313 has been received after more than an exemplary 10 seconds after the start of the timer in step 308c (e.g. maximum exemplary time between message 122 and message 123), then step 324 comprises responder 102' caching the data from message 313 in memory 102f until responder 102' receives a retry message 122 again. Upon responder 102' receiving the message 122 again (this time after message 313 has been received with data from message 313 cached in memory 102f), then responder 102' can send the cached data from message 313 recorded in the step 324.

Using the data received from DPP server 103, responder 102' can then send initiator 101 a DPP authentication response message 123. In exemplary embodiments, message 123 is sent as a unicast message from (i) the responder MAC address MAC.R 102n (which received message 122) to (ii) the initiator MAC address MAC.I 106n (which sent message 122), over (iii) WiFi link 192. As

depicted in Figure 3a, message 123 can include hash values of the bootstrap public keys H(Br 102a), H(Bi 101a), the responder ephemeral public key Pr 102p, and the ciphertext 315c. The ciphertext 315c can have encrypted values for (i) R-nonce 318b, (ii) I-nonce 317a, (iii) R-capabilities 318c, and (iv) ciphertext 324a. The ciphertext 315c can be encrypted with mutually derived symmetric encryption key k2 314c. Ciphertext 324a can be encrypted with the mutually derived symmetric key ke 327a and can include a plaintext value for R-auth 404. As described above, R-auth 404 can comprise a hash value over an initiator nonce 317a received and the responder nonce 318b.

As depicted in Figure 3a, initiator 101 can receive DPP Authentication Response message 123. Although not depicted in Figure 3a, in exemplary embodiments, initiator 101 can conduct a step 308b to process and validate the received responder ephemeral public key Pr 102p, such as evaluating that the received key in message 123 is securely compatible with key pi 101d and key bi 101b, using the same steps as a key validation step 308c. In other words, a key validation step 308c by responder 102' may not be as complete as a key validation step 308b by initiator 101, since initiator 101 can compare the received key Pr 102p from a message 123 with data from private keys pi 101d and key bi 101b using parameters 199a for the keys.

Initiator 101 can receive DPP authentication response message 123 from responder 102' and conduct a series of steps in order to process the message and reply to the responder 102'. At step 325, initiator 101 can record the validated, received responder ephemeral public key Pr 102p in a responder public key table 101ta, as depicted in Figure 1d above. In other words, although Figure 1d does not depict a column for key Pr 102p, the column and data could be added for a responder as the data is received. Initiator ephemeral private key pi 101d and initiator ephemeral public key Pi 101c could be recorded in a responder public key table 101ta as well, when the keys can be utilized. Any insertion of key Pr 102p, pi 101d and Pi 101c (for responder 102' identified by key Br 102a into table 101ta preferably has the data deleted upon conclusion of a DPP 191 and receipt and installation of credentials 109 as a configuration object for device 106 with initiator 101.

At step 319a, initiator 101 can conduct a second initiator key exchange step 319a, where initiator 101 can input the received, validated responder ephemeral public key Pr 102p and the derived initiator ephemeral private key pi 101d into an ECDH key exchange algorithm 401b in order to calculate a shared secret key N 411. The second server key exchange step 319a is also depicted and described in connection with Figure 4b below. Shared secret key N 411 can also be used by initiator 101 in a step 327 below. Initiator 101 can input shared secret key N 411 into a key derivation function 402b in order to mutually derive the symmetric encryption key k2 314c.

At step 320b, initiator 101 can use the derived key k2 314c from a step 319a in order to conduct a decryption step 320b on received ciphertext 315c. A decryption step 320b on received ciphertext 315c using key k2 314c is depicted and described in connection with Figure 4b below. Initiator 101 can read the plaintext 403b values from decryption step 320b. The plaintext values 403b can include

the initiator nonce 317a from a message 122 and message 309 above. The plaintext values 403b can also include a second ciphertext 324a (e.g. ciphertext 324a can be inside ciphertext 315c). Initiator 101 verifies that the initiator nonce 317a received in message 123 and decrypted in step 320b is equal to the initiator nonce 317a value sent in message 122 above. If the two initiator nonce 317a values are not equal, then initiator 101 can abort the DPP procedure and subsequent steps.

At step 327, initiator 101 can conduct a third initiator key exchange step 327, where initiator 101 can determine the secret shared key L 412 by inputting (i) the an ECC point addition of the responder bootstrap public key Br 102a and the responder ephemeral public key Pr 102p, and (ii) the initiator bootstrap private key bi 101b into an ECDH key exchange algorithm 401c. This third server key exchange step 327 is depicted and described in connection with Figure 4b below. The shared secret key L 412 along with share secret keys M 410 and N 411 can be used with a key derivation function 402c in order to output the mutually derived symmetric ciphering key ke 327a.

At step 328, initiator 101 can conduct a decryption step 328 on the ciphertext 324a using the key ke 327a. A decryption step 328 is depicted and described in connection with Figure 4c below. Initiator 101 can use the decryption step 328 in order to read the plaintext value of R-auth 404. R-auth 404 can comprise a hash 199e value over at least a combination of initiator nonce 317a and the responder nonce 318b. At step 328, initiator 101 can then also internally calculate the same value for R-auth 404, and if the internally calculated value and the received, decrypted value are equal, then the responder 102' can be consider authenticated with initiator 101. The calculation of R-auth 404 by initiator 101 can be according to section 6.2.3 of DPP specification version 1.0.

Initiator 101 can then conduct a step 322b, where step 322b can comprise calculating an initiator authentication value I-auth 404a. Initiator authentication value 404a can be similar to responder authentication value R-auth 404, except the values of initiator nonce 317a can be sequenced before a responder nonce 318b inside a hash 199e calculation over the two nonces. An initiator 101 can then encrypt the initiator authentication value 404a using an encryption step 322 using key ke 327a, as depicted in Figure 4c below. In other words, an encryption step 322 in Figure 4c depicts the encryption of R-auth 404, and encryption step 322b can comprise a initiator 101 encrypting the initiator authentication value I-auth 404a using the key ke 327a derived in a step 327. The resulting ciphertext with the encrypted initiator nonce valued 404a from an encryption step 322b can comprise ciphertext 324b. As depicted in Figure 3a, the collection of steps for an initiator 101 of the depicted steps 325, 319a, through 322b can comprise a step 331. A step 331 in summary allows initiator 101 to process and verify the received DPP authentication response message 123. As depicted in Figure 3a, additional steps for a responder 102' in initiator 101, and DPP server 103 to conduct in order to complete a device provisioning protocol session are depicted in Figure 3b below.

Figure 3b

Figure 3b is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a DPP server, a responder proxy, and an initiator, in accordance with exemplary embodiments. Before starting steps and message flows depicted in Figure 3b, the device 106 with initiator 101, responder 102', and DPP server 103 can complete the series of steps and message flows depicted above in Figure 3a. System 300b can include a DPP server 103, responder proxy 102', and initiator 101 operating in device 106. Responder proxy 102' can communicate with DPP server 103 via IP network 113 as depicted in Figure 1a. Responder proxy 102' can communicate with initiator 101 via WiFi link 192.

After completing the series of steps for a step 331 in Figure 3a, initiator 101 can send responder 102' a DPP authentication confirm message 123a. The data and steps to determine and calculate a DPP authentication confirm message 123a can be compatible with section 6.2.4 of DPP specification version 1.0. Note that although the DPP authentication confirm message 123a can be equivalent to the DPP authentication confirm message from section 6.2.4 of DPP specification version 1.0, an important difference can be the operation of responder 102' in Figure 3b versus a responder as contemplated by DPP specification version 1.0. In the present disclosure as depicted in Figure 3b, responder 102' does not record the responder bootstrap private key br 102b. However, using the technology depicted in Figure 3a and Figure 3b, a responder 102' can (i) send and receive messages that are fully compatible with a DPP protocol 191 (including DPP specification version 1.0), but (ii) DPP server 103 can record and operate on key br 102b. In other words, a initiator 101 may not "know" that a responder 102' operates as either (i) an responder 102* from Figure 1b or Figure 1c, or (ii) an responder 102' from Figure 3a and Figure 3b since the messages transmitted and received by the initiator 101 from responder 102' can be fully compatible with a DPP 191.

Other differences besides the location of recording and operating on key br 102b can apply as well for the responder mode 306a depicted in Figure 3a and Figure 3b, such as key pr 102r also remains in DPP server 103 and not recorded or operated on by responder 102'. The recording and operation of responder private keys br 102b and key pr 102r by DPP server 103 can comprise a first responder mode 306a, where responder mode 306a was selected by DPP server 103 or network 107 in a step 306 and communicated to responder 102' in a message 312b.

Responder 102' can listen for either (i) a broadcast message with the hash 199e values for the bootstrap public keys 102aa and 101aa, or (ii) a unicast message to MAC.R 102n from MAC.I 106n for initiator 101 operating in device 106. Responder 102' can receive the DPP authentication confirm message 123a and process the message. DPP authentication confirm message 123a can be processed by initiator 101 using a step 331 as depicted in Figure 3a. DPP authentication confirm message 123a

can include an encrypted value for initiator authentication value 404a as a ciphertext 324b, where ciphertext 324b is encrypted with derived key ke 327a in a step 322b.

Upon receipt of DPP authentication confirm message 123a from device 106, responder 102' can forward data in the message to DPP server 103 via secure session 302b. Responder 102' can send DPP server 103 a message 329, where message 329 includes an identity for device 106 (which could be the depicted value of MAC.I 106n) as well as ciphertext 324b. Although not depicted for a message 329 in Figure 3b, ciphertext 324b can also include the hash values for the bootstrap public keys of hash value 102aa and/or hash value 101aa.

DPP server 103 can receive message 329 and record data from the message 329 and process the data. DPP server 103 can use the identity for device 106 comprising MAC.I 106n (or another device identity such as ID.device 106i or hash value 101aa sent in a message 329). DPP server 103 can first conduct a decryption step 328b, where decryption step 328b can use the same steps as a decryption step 328 as depicted in Figure 4c below, with a difference being decryption step 328b operates on the ciphertext 324b containing the encrypted initiator authentication value I-auth 404a instead of the R-auth 404 value. After a step 328b, DPP server 103 can read the plaintext initiator authentication value I-auth 404a.

In step 330, DPP server 103 can compare the decrypted value for I-auth 404a with a calculated value for I-auth 404a and if the two values match then the initiator 101 can be considered by DPP server 103 to be authenticated. The calculation of I-auth 404a for DPP server 103 in a step 330 can comprise a calculation of a secure hash value over at least the responder nonce 318b and bootstrap and ephemeral public keys, with an exemplary calculation for I-auth 404a shown in the sixth paragraph of page 50 of DPP specification version 1.0. Other calculations of an initiator authentication value over the responder nonce 318b are possible as well, without departing from the scope of the present disclosure.

In a step 333, responder 102' can select a set of network credentials 109 for device 106 using initiator 101. The set of network credentials 109 could comprise the exemplary set of credentials 109 depicted and described in connection with Figure 1a. The set of network credentials 109 can comprise an SSID.network-AP 109a, a pairwise master key (PMK) PMK.network-AP 109b, and a set of configuration data comprising config.network-AP 109c. The value PMK.network-AP 109b can also include an identifier for a PMK such as a PMKID. Or, for legacy WiFi connections that utilize WPA2 and similar or earlier standards, PMK.network-AP 109b can comprise a value for a shared secret key such as a PSK. The set of network credentials 109 can comprise values for device 106 to authenticate and encrypt data with network access point 105 in order for device 106 to connect with network access point 105. The network credentials 109 could support use of WiFi link 192 with WPA3 standards from the WiFi Alliance TM in exemplary embodiments. Note that in some exemplary embodiments, a step 333 for selecting credentials 109 for device 106 can be omitted by responder 102' and the step

333 can be performed by DPP server 103 or another server within network 107, such as a server running a RADIUS server, or performing an extensible authentication protocol (EAP) based authentication with device 106.

In a step 334, for embodiments where responder 102' conducts the step 333 to select credentials 109, responder 102' can optionally encrypt the credentials 109 with an asymmetric ciphering algorithm using a public key for initiator 101 or device 106. The use of an asymmetric ciphering algorithms and associated public keys is depicted below in a step 334 in Figure 3e. Responder 102' can send DPP server 103 a message 335, where the message 335 includes an identity for device 106 such as MAC.I 106n, and a ciphertext 334a, where the ciphertext includes the asymmetrically encrypted network credentials 109'. The encryption key depicted in Figure 3b for ciphertext 334a is initiator ephemeral public key P_i 101p, although a different public key for device 106 or initiator 101 could be used as well, such as the initiator bootstrap public key or a public key in certificate cert.device 106m. The device certificate cert.device 106m could be (i) stored in a device database 103d and (ii) sent to DPP server 103 in a message 312a in Figure 3a, and then (ii) sent from DPP server 103 to responder proxy 102' in a message 312b in Figure 3a.

The "optional" notation for a message 335 depends on the source of credentials 109 in a system 300b. If a device operating responder 102' (such as an exemplary access point 105) is responsible for generating or operating credentials 109 (such as a single access point in a user's home), then responder 102' can be the source of the credentials 109 and send the message 335. If a network 107 is responsible for generating or operating credentials 109 (such as in a corporate or campus network of LAN access points 105), including operating EAP-based authentication via RADIUS or similar standards, then the message 335 can be optionally omitted since DPP server 103 could obtain the network credentials 109 from another source in a network 107 besides the device operating responder 102'.

After transmitting the DPP authentication confirm message 123a, in step 332, initiator 101 can take on a roll of enrollee, such that initiator 101 can prepare to send a configuration attribute 330a. Configuration attribute 330a can be used to specify that initiator 101 can take the roll of enrollee and is expecting a set of network credentials 109 from responder 102'. A configuration attribute 330a can be formatted according to section 6.3.4 of DPP specification version 1.0. For the exemplary embodiment depicted in Figure 1a, device 106 with initiator 101 would prefer a role of WiFi client (or "sta") for a configuration attribute 330a, but other possibilities exist as well. For the exemplary embodiment depicted in Figure 7 below a responder 102' operating within an access point 105 would prefer a role of WiFi access point (or "ap") for a configuration attribute 330a.

In step 332, initiator 101 can also generate a third random number, for an enrollee nonce E-nonce 332a. E-nonce 332a and other nonce values in Figure 3a such as the initiator nonce I-nonce 317a and responder nonce R-nonce 318b can be processed using nonce parameters 199g. Initiator 101 can then conduct an encryption step 322c, where encryption step 322c can comprise (i) an encryption

step 322 using key ke 327a, and (ii) the plaintext to be encrypted can be at least the E-nonce 332a and the configuration attribute 330a. The output of an encryption step 322c can be a ciphertext 324c. Initiator 101 can then send responder 102' a DPP Configuration Request message 123b, where the configuration request 123b can include the ciphertext 324c, where the plaintext in ciphertext 324c can include E-nonce 332a and the configuration attribute 330a. DPP Configuration Request message 123b can be processed by initiator 101 according to section 6.3.3 of DPP specification version 1.0.

Responder 102' can receive DPP Configuration Request message 123b via connection 192 and using DPP application 102g. As contemplated herein, a DPP application 102g can perform the steps for responder 102' depicted in Figure 3a, such as transmitting/receiving DPP 191 messages using a first MAC.R 102n for connection 192 and transmitting/receiving messages with DPP server 103 using MAC.N 102q via access network 112. DPP application 102g could also perform the depicted steps 308b above to validate the received initiator ephemeral public key Pi 101c. For embodiments where responder 102' uses a WAN radio 102h to connect with IP network 113 and DPP server 103, then DPP application 102g could also use a different identity than MAC.N 102q to identify the physical interface of WAN radio 102h, such as an IMEI, or other possibilities exist as well. For other embodiments MAC.R 102n could also potentially be the same as MAC.N 102q, and other possibilities exist as well for the identification of physical network interfaces used by responder 102' or a device operating responder 102' without departing from the scope of the present disclosure.

Responder 102' can send DPP server 103 a message 336 via secure connection 302b, where message 336 can include the ciphertext 324c. Message 336 can also include an identity for device 106, such as, but not limited to, MAC.I 106n or ID.device 106i, since both DPP server 103 and responder 102' can communicate with a plurality of different devices 106 over time, and DPP server 103 may prefer to know which ciphertext 324c is associated with which device 106 in order to process the ciphertext 324c. As depicted in Figure 3b, message 336 can include the encrypted values for E-nonce 332a and configuration attribute 330a.

DPP server 103 can receive message 336 and process the message 336. DPP server 103 can conduct a decryption step 328c using decryption step 328 depicted in Figure 4c below. A difference between decryption step 328 and decryption step 328c is that decryption step 328c uses the ciphertext 324c as input into the symmetric decryption algorithm 405b. DPP server 103 can use the previously calculated key ke 327a in order to read the plaintext within ciphertext 324c. In a step 328c, DPP server 103 can record the plaintext E-nonce 332a and configuration attribute 330a.

In step 337, DPP server 103 can process the received configuration attribute 330a. For the exemplary embodiment depicted in Figure 1a, configuration attribute 330a can specify that device 106 with initiator 101 may have a role of WiFi client. For a step 337, DPP server 103 can select a set of network credentials 109 for device 106 using initiator 101 in order to create a configuration object. The configuration object in a step 337 can be compatible with a configuration object in section 6.3.6

of DPP specification version 1.0. For embodiments where DPP server 103 received a message 335 with an asymmetrically encrypted set of credentials 109' from responder 102', then in a step 337 DPP server 103 can select and use the credentials 109' as the data for a configuration object. If encrypted credentials 109' are selected by DPP server 103 in a step 337, then the credentials 109' after decryption by device 106 using a decryption step 339 can comprise a configuration object per section 6.3.6 of DPP specification version 1.0. As contemplated herein, a reference to DPP specification version 1.0 can also include subsequent and related versions of the standard that would be substantially compatible with version 1.0.

Continuing at step 337, for embodiments where DPP server 103 did not receive a message 335 from responder 102', then DPP server 103 can be responsible for obtaining credentials 109, such as if network 107 is responsible for generating or creating the network credentials 109 for device 106 (such as if access point 105 uses EAP-based authentication and other possibilities exist as well). Note that DPP server 103 could also receive an asymmetrically encrypted set of network credentials 109' from another server or external network 108 in a step 337, where DPP server 103 or network 107 sends an identity for device 106 such as Bi 101a or H(BI) 101aa to the external network 108 and the external network 108 can use the identity 106 to conduct a step 334 for credential 109'. Or, in a step 337, DPP server 103 could receive plaintext network credentials 109 from another server in network 107 or generate the network credentials 109 using data recorded for access point 105 in a server database 103a. Other possibilities exist as well for the source of a set of network credentials 109 for DPP server 103 in a step 337 without departing from the scope of the present disclosure.

Continuing at step 337, DPP server 103 can process a configuration object for network credentials 109, and the configuration object can include an SSID value of SSID.network-AP 109a, a pairwise master key value of PMK.network-AP 109b, and configuration values of config.network-AP 109c. The value of PMK.network-AP 109b can also include a value for PMKID (e.g. PMK identifier), such that PMK.network-AP 109b is used by each of a client and an access point with the PMKID value. The configuration values of config.network-AP 109c in a configuration object could also specify required supporting data for operation of device 106 with an access point 105 after a configuration step 106a, such as an operating class 131a, channel list 131b, an authentication mode for device 106 to operate, which could be PSK, PMK, SAE, EAP, also with any required supporting data and/or identities in order to use the authentication mode. The configuration values of config.network-AP 109c in a configuration object could also include a passphrase or pre-shared key values of PSK for legacy WPA2 networks. In other words, credentials 109 for other authentication schemes between a device and an access point could be supported as well, including EAP-TLS, EAP-PSK, EAP-AKA, EAP-AKA', etc.

As contemplated herein, a set of network credentials 109 can include either (i) network credentials 109 for a selected primary access network, and/or (ii) network credentials 109 for both a

selected primary access network and a selected backup access network. Or, in another exemplary embodiment, although a single configuration object of network credentials 109 is depicted in Figure 3b, DPP server 103 could process (i) a first set of network credentials 109 for selected primary access network into a first configuration object and then (ii) a second set of network access credentials 109 for selected backup access network into a second configuration object. At the conclusion of a step 337, DPP server 103 can record and process a configuration object comprising a set of plaintext network credentials 109 or asymmetrically encrypted set of network credentials 109' for device 106 operating initiator 101, where device 106 can be identified by an identity such as ID.device 106i or MAC.I 106n or H(Bi) 101aa.

For a step 338 in Figure 3b, DPP server 103 can conduct a signature creation step 338 as depicted and described in connection with Figure 3d below. The configuration object with network credentials 109 (or encrypted credentials 109') can be signed using the recorded responder ephemeral private key pr 102r, corresponding to public key Pr 102p sent in message 325. The digital signature by DPP server 103 in Figure 3b can comprise a digital signature 338c. Or, in another embodiment the configuration object with network credentials 109 can be signed using the recorded responder bootstrap private key 102b, where initiator 101 could verify the signature 338c with the bootstrap public key Br 102a. Note that some exemplary embodiments contemplate that DPP server 103 receives (i) credentials 109 or asymmetrically encrypted credentials 109' from (ii) another server in network 107 or an external network 108, and a step 338 can comprise DPP server 103 creating a digital signature 338c for the (i) credentials 109 or asymmetrically encrypted credentials 109' from (ii) another server in network 107 or an external network 108. The steps to create and verify a signature with ECC PKI keys can be according to FIPS publication 186-4, and other possibilities exist as well. The use of a digital signature 338c for credentials 109 or 109' is optional in exemplary embodiments, and for some embodiments a digital signature 338c could be omitted (as well as signature creation step 338 and signature verification step 340).

At step 332d, DPP server 103 can conduct an encryption step 332d using an encryption step 322 depicted and described in connection with Figure 4c below. Encryption step 332d can be performed using key ke 327a for device 106. For encryption step 332d, the plaintext values of (i) the E-nonce 332a and (ii) the configuration object comprising network credentials 109 (or asymmetrically encrypted credentials 109') can be input into a symmetric encryption algorithm 405a in order to generate a ciphertext 324d. Note that digital signature 338c can be included in an encryption step 332d as well, although digital signature 338c could also optionally not be included in a ciphertext 324d and sent in a subsequent message 341 and 124 external to ciphertext 324d. As depicted in Figure 3b, the collection of steps 328c through 322d can comprise a step 349.

DPP server 103 can then send responder 102' a message 341, where message 341 includes the ciphertext 324d. Message 338 can also include (i) an identity for device 106 such as MAC.I 106n or

ID.device 106i, and (ii) a signature 338c from a signature creation step 338. Message 338 can be sent via secure connection 302b, as depicted in Figure 3b. Although signature 338c is depicted as within ciphertext 324d for a message 341, signature 338c could be external to ciphertext 324d in a message 341.

5 Responder 102' using DPP application 102g can receive message 341 via secure connection 302b. Responder 102' can use an optional identity for device 106 in a message 341 in order to determine which of a potential plurality of devices 106 is associated with ciphertext 324d in order to forward the data to device 106. Responder 102' can then send initiator 101 a DPP configuration response message 124 via connection 192, where message 124 can include the ciphertext 324d, and
10 the ciphertext 324d can include the E-nonce 332a, the signature 338c, and either network credentials 109 or asymmetrically encrypted network credentials 109'. If network credentials 109' are sent in ciphertext 324d, then the network credentials 109' can comprise a set of "double encrypted" network credentials of (i) asymmetrically encrypted network credentials 109' and (ii) then encrypted again in a ciphertext 324d using the key ke 327a.

15 Initiator 101 can receive message 124 from responder 102' with ciphertext 324d. Initiator 101 can conduct a decryption step 328d using decryption step 328 depicted in Figure 4c below. A difference between decryption step 328 and decryption step 328d is that decryption step 328d uses the ciphertext 324d as input into the symmetric decryption algorithm 405b. Initiator 101 can use the previously calculated key ke 327a in order to read the plaintext within ciphertext 324d. Initiator 101
20 can confirm that plaintext in ciphertext 324d includes the previously sent E-nonce 332a. Initiator 101 can extract a plaintext value for (i) network credentials 109 or (ii) asymmetrically encrypted network credentials 109' using symmetric decryption algorithm 405b and key ke 327a. In other words, with credentials 109' decryption step 328d removes the first layer of "double encryption" where the first layer was encrypted with key ke 327a, and the "plaintext" after decryption step 328d for credentials
25 109' can comprise the asymmetrically encrypted network credentials 109' (e.g. with the first layer of encryption with key ke 327a has been removed).

Initiator 101 can conduct a signature verification step 340 as depicted and described in connection with Figure 3d above in order to verify signature 338c received in message 124. Initiator 101 can use either (i) responder ephemeral bootstrap public key Pr 102p or (ii) responder bootstrap
30 public key Br 102a in order to conduct a signature verification step 340 on digital signature 338c. If plaintext network credentials 109 are included in ciphertext 324d, then initiator 101 can read the plaintext values for credentials 109. If asymmetrically encrypted network credentials 109' are included in ciphertext 324d, then initiator 101 or device 106 operating initiator 101 can conduct an asymmetric decryption step 339 depicted below in Figure 3e in order to convert credentials 109' into
35 plaintext credentials 109.

After reading the plaintext network credentials 109 and optionally verifying a signature using recorded public keys 102p or 102a, device 106 can complete the configuration step 106a by loading the network credentials 109 and begin operating a WiFi radio 106r with the network credentials 109. In other words, the WiFi data link 192 depicted in Figure 1a can be converted into a fully authenticated and encrypted WiFi connection using network credentials 109 and standards such as WPA2, WPA3, and also subsequent or related standards where WiFi nodes such as clients or access points use a compatible set of network credentials 109. For example, device 106 could begin scanning for SSID.network-AP 109a on a channel in Config.network-AP 109c, and then send access point 105 a probe request. In this manner, device 106 can obtain connectivity to network access point 105.

Figure 3c

Figure 3c is an illustration of an exemplary server database, with tables for a server database recording exemplary data, in accordance with exemplary embodiments. A server database 103a depicted and described above in connection with Figure 1a and Figure 3a can record data for a DPP server 103 to work with a responder 102' in order for the responder 102' to conduct a device provisioning protocol 191 with a device 106 operating an initiator 101. A server database 103a could record a plurality of tables for a plurality of devices 106 and a plurality of responders 102', including the depicted device credentials table 345. Other possibilities exist as well for the organization, tables, and recorded data within a server database 103a without departing from the scope of the present disclosure. Data within server database 103a could be encrypted using a symmetric key. As depicted in Figure 1a, a DPP server 103 operating in a network 107 could operate and record a server database 103a.

Server database 103a can record a device identity for device 106 of MAC.I 106n, a responder identity which could be MAC address MAC.N 102q, a responder mode 306a comprising a mode of operation for the DPP server 103 and responder 102', a selected set of Cryptographic Parameters 199a, a Responder Ephemeral Public Key Pr 102p, a Responder Ephemeral Private Key pr 102r, an Initiator Ephemeral Public Key Pi 101c, Bootstrap keys 399, an optional PKEX Key 198, a Shared Secret k1 314a, a Shared Secret k2 314c, and a Shared Secret ke 327a. Different identities for device 106 and responder 102' could be recorded in a server database 103a as well, such as ID.device 106i or an identity for a responder 102' that is different than a MAC address for responder 102', such as a serial number or unique identifier for responder 102'. In exemplary embodiments, hash value 101aa over the initiator bootstrap public key Bi 101a can be used for a device identity in a server database 103a.

For exemplary embodiments, not all depicted values may be recorded in a server database 103a, as shown by the depicted value of "--", which can mean that the value is not recorded or stored in a server database 103a. A responder 102' could record and operate on the value depicted as "--" for

a server database 103a in Figure 3c for some responder modes 306a. For example, with the row in server database 103a depicted with a mode “Fig. 5a”, the responder 102’ could derive, record, and operate on the responder ephemeral public key Pr 102p, and DPP server 103 may not need or record the key. Similarly, with the row in server database 103a depicted in Figure 3c as a mode of “Fig. 5a”, the responder 102’ could derive or operate on shared secret keys k1 314a, k2 314c and ke 327e, while server database 103a would not need to record or operate on the shared secret keys in order for responder 102’ with a mode 306a of “Fig. 5a” to conduct a DPP 191 with initiator 101.

Note that in alternative exemplary embodiments, a server database 103a could record the values M 410, N 411, and L 412 respectively instead of key k1 314a, k2 314c, and/or ke 327e, and either server 103 or responder 102’ could use the key derivation function 402a, 402b, and/or 402c, respectively in order to convert the values M, N, and L into keys k1, k2, and ke as depicted in Figures 4a through 4c. In exemplary embodiments, the time that a device shared secret key or a derived ephemeral PKI key, or a bootstrap key is recorded in a server database 103a can be minimized to the minimum time necessary to conduct a DPP 191. For example, shared secret keys and ephemeral PKI keys recorded in a server database 103a can be deleted once a configuration step 106a has been completed for a device 101, such as at the time DPP server 103 receives a confirmation that device 106 has begun using the credentials 109 with a network access point 105.

Responder mode 306a can specify an operating mode for responder 102’ and DPP server 103, such that a sequence of steps and transmitting/receiving PKI and/or shared secret keys and/or related supporting messages can be specified by the responder mode 306a. A responder mode 306a could also be referred to as a “server mode” 306a, since the mode specifies the operation of a DPP server 103. Or, responder mode 306a could be referred to as a “mode 306a”, since both steps for responder 102’ and DPP server 102 are affected by the mode. Other terms could be used to refer to the selected mode as well, such as a “method”, “procedure”, or a “process”. Figure 2a above also described a first, second, and third operating responder modes 306a. A first mode is depicted as “Fig. 3” in Figure 3c and can comprise the mode 306a depicted in Figure 3a and Figure 3b. Note that all depicted ephemeral PKI keys and bootstrap PKI keys, as well as all depicted shared secret keys can be recorded in a server database 103a for a responder 102’ for the first mode. In other words, when conducting a DPP 191, responder 102’ and DPP server 103 could have all the depicted keys recorded in a server database 103a for the first mode. This first mode for an initiator mode 306a can be preferred for embodiments where (i) responder 102’ may have a lower level of trust with network 107, or (ii) responder 102’ could have a lower level of processing resources, or (iii) an owner for device 106 or a network 107 may prefer that responder bootstrap private key br 102b is not shared externally to network 107, (iv) regulatory requirements where device 106 could comprise a high value or sensitive piece of equipment, or (v) responder 102’ conducts a secure session 302b without a responder certificate cert.responder 102m. Other possibilities exist as well for the reasons different modes 306a may be

used in a server database 103a to specify the steps for a responder 102' and DPP server 103 to conduct a DPP 191 with device 101.

A second responder mode 306a is depicted as "Fig. 5a" in Figure 3c, where (a) the steps and message flows in Figure 5 for responder 102' and DPP server 103 can comprise (b) the second mode.

Note that all relevant bootstrap PKI keys 399 are recorded in a server database 103a for the second responder mode 306a depicted in Figure 5a. The second mode 306a can also omit (i) the initiator ephemeral public key P_i 101c, (ii) and the responder ephemeral private key pr 102r from being recorded in a server database 103a. Note that secret shared keys can optionally not be recorded in a server database 103a for the second mode depicted in Figure 5a. This exemplary embodiment of a second responder mode 306a may be preferred for cases where (i) responder 102' can handle more of the cryptographic operations (e.g. reducing DPP server 103 processing load), but (ii) a network 107 could prefer that the responder bootstrap private key br 102b is not transmitted to or recorded in responder 102'. A third responder mode 306a is depicted as "Fig. 5b" in Figure 3c, where (a) the steps and message flows in Figure 5b for responder 102' and DPP server 103 can comprise (b) the third mode.

A fourth responder mode 306a in Figure 3c can comprise responder 102' operating as a responder 102* depicted in Figure 1b and Figure 1c, and in this case responder 102* can record the keys shown for the third mode 306a in Figure 3b (depicted as "102*"). A DPP server 103 can still provide the bootstrap keys 399 shown to responder 102* in this third mode of operation, or the responder 102* could obtain the bootstrap keys 399 from other means, such as "out of band" messages 121 and 126 in Figure 1c. In addition, a responder 102* could obtain the bootstrap keys "in band" (e.g. via WiFi link 192) using PKEX key 198. In exemplary embodiments, a fourth responder mode 306a where responder 102' operates as a responder 102* as depicted in Figure 1b and Figure 1c (e.g. responder 102' would record responder bootstrap private key br 102b), responder 102 can conduct a step 604 as depicted in Figure 6 in order to obtain credentials 109 from a DPP server 103. In other words, although the conventional technology depicted for a responder 102* in Figure 1b and Figure 1c contemplates sending credentials 109 from responder 102* to initiator 101, the conventional technology does not suggest responder 102* obtains the credentials from a DPP server 103. A step 604 as depicted and described below in connection with Figure 6 could be used by a responder 102' with the fourth responder mode 306a in order for responder 102' (operating as a responder 102*) in order to obtain credentials 109 or credentials 109' from a DPP server 103.

Bootstrap keys 399 in a server database 103 can record the bootstrap keys depicted in Figure 3c. For embodiments where a device 106 can support multiple different sets of cryptographic parameters 199a, a server database 103a can record the selected set of cryptographic parameters 199a for communication between a responder 102' and device 106 with initiator 101. The keys for "bootstrap keys 399" can be according to the selected set of parameters 199a. In other words, a device

database 104 in Figure 2a above can record all versions of the bootstrap PKI keys for all supported cryptographic parameters 199a for a device 106. However, since a set of cryptographic parameters 199a can be selected for a DPP 191 (e.g. based on data received from initiator 101 in message 122 such as key Pi 101c or hash value 102aa), then the bootstrap keys 399 for that selected set of cryptographic parameters 199a can be recorded in a server database 103a.

A server database 103a can also record a pre-shared PKEX key 198, if PKEX is optionally supported by initiator 101. PKEX key 198 can be used by responder 102' to securely transfer bootstrap public keys in band (i.e. via connection 192 in Figure 1a and Figure 3a). The use of a PKEX protocol to exchange the public keys is described in section 5.6 of DPP specification version 1.0. For embodiments where responder 102' uses PKEX key 198 with initiator 101, the responder bootstrap private key br 102b preferably continues to reside within a network 107 and can be in a server database 103a. Embodiments can omit the use of PKEX key 198, as depicted with the value of "NA" for the key in Figure 3c. Derived, shared secret keys with initiator 101 can also be recorded in a server database 103a. The derived shared secret keys could comprise key k1 314a, key k2 314c, and key ke 327e. The steps to derive and use these keys are depicted and described below in Figure 4a, Figure 4b, and Figure 4c, respectively. As discussed above, some keys may reside in server database 103a, and other keys may reside within a responder 102' (depicted with a "-"), depending on the responder mode 306a.

The present disclosure contemplates other implementations of a device provisioning protocol besides the DPP specification version 1.0. These other implementations could have different mutually derived shared secret keys than those depicted in Figure 3c, and the different mutually derived shared secret keys could be recorded in a server database 103a as well. For example, different key derivation functions 402 could be used to derive a secret shared key than that specified in DPP specification version 1.0. In addition, other or different PKI keys for a responder 102' and/or initiator 101 could be recorded in a server database 103a. For example, the DPP specification version 1.0 could be extended to explicitly include a DPP server 103 public and private key, in order to extend support with a DPP server 103, and in these embodiments the DPP server 103 public and private keys could be included in a server database 103. Other possibilities exist as well for a server database 103a to record PKI keys and shared secret keys without departing from the scope of the present disclosure.

A server database 103a could also record a device credentials table 345. Device credentials table 345 can record a device identity ID.device 106i, MAC.I 106n, a set of primary network credentials 350a, and a set of backup network credentials 350b. The set of primary network credentials 350a can comprise the set of credentials (plus configuration) for a device 106 to use with a selected primary network, such as an exemplary network access point 105 in Figure 1a. The set of backup network credentials 350b can comprise the set of credentials (plus configuration) for a device 106 to use with a selected backup network. The credentials in a device credentials table 345 could be obtained

from during a step 333 or a step 337 as depicted and described above in connection with Figure 3b. Note that a DPP server 103 could conduct a step 337 in order to select the network credentials 109, including receiving the credentials 109 from an external network 108. DPP server 103 could receive and record the asymmetrically encrypted network credentials 109' in a device credentials table 345.

5 For example, with the first device depicted in a device credentials table 345, the credentials in device table for the first device depicted could comprise a set of plaintext credentials 109 depicted in Figure 1a. The depicted designation of "109-1" can identify the specific set of plaintext credentials 109 for a primary network 350a device 106 with ID.device 106i. The device in the second row can use the same plaintext network credentials 109. The credentials stored for the primary network for
10 the third device can comprise depicted credentials "109' -1", which can mean asymmetrically encrypted network credentials 109' are recorded. The asymmetrically encrypted network credentials 109' could be received from an external network 108 (or possibly just securely recorded by a network 107), and only the third device could feasibly decrypt the credentials 109' using a step 339 and a recorded private key for the device 106.

15 In exemplary embodiments, the asymmetrically encrypted credentials 109' are encrypted using Elgamal encryption and either (i) the initiator bootstrap public key Bi 101a or (ii) the initiator ephemeral public key Pi 101c. The DPP server 103 could send either (i) the initiator bootstrap public key Bi 101a or (ii) the initiator ephemeral public key Pi 101c (along with an identity for device 106 such as ID.device 106i or MAC.I 106n) to a server that conducts an asymmetric encryption step 334
20 on plaintext credentials 109. The DPP server 103 could receive the encrypted credentials 109' and record them for device 106 with ID.device 106i in a device credentials table 345. Additional details for receiving credentials from an external network 108, and recording the encrypted credentials 109' in a server database 103 are depicted and described below in connection with Figure 3b, depicted as credentials 109'.

25 Note the present disclosure contemplates that (a) credentials 109 or 109' could utilize wireless networking technology other than WiFi or 802.11 based technology. For example, surrounding wireless networks for device 101 could use exemplary potential 5G wireless technology or LPWAN networking technology potential wireless networks for device 106. Thus, credentials 109 or 109' in a server database 103a and used by device 106 can support networking technology other than WiFi. As
30 one example, an initiator 101 could be operated by an Ethernet interface on a computer or server, and an Ethernet switch could operate a responder 102'. The flow of messages for a DPP 191 protocol could be conducted via a physical wired Ethernet link instead of a WiFi link 192. In this manner, the device operating an initiator 101 could be authenticated and configured by an Ethernet switch operating the responder 102'. As one example, the values and configuration in a dynamic host configuration protocol (DHCP) could be sent as credentials 109 or credentials 109' in a message 124
35 from an Ethernet switch operating a responder 102'.

In another exemplary embodiment, a wireless WAN network, such as a 5G or 6G wireless network according to future 3GPP or ETSI standards, could implement support for the device provisioning protocol contemplated herein. A base station or base transceiver station for the wireless network could operate as a responder 102' and a user device desiring to connect with the network could operate as an initiator 101. The user device and base station could conduct a DPP 191 and the base station could send credentials 109 or credentials 109' for the 5G or 6G wireless network to the user device in a DPP message 124 after successful authentication using DPP. The credentials 109 or 109' for either a wired Ethernet network or a wireless WAN network could be recorded for a device 106 or a plurality of devices 106 in a server database 103a. The use of a primary and backup network for device 101 can provide increased reliability for the operation of device 106, such that if the primary access network for credentials 350a is not available then device 101 could use the selected backup network with credentials 350b in order for device 106 to maintain connectivity to an IP network 113.

Figure 3d

Figure 3d is a flow chart illustrating exemplary steps for creating and verifying a digital signature using PKI keys, parameters, and data input, in accordance with exemplary embodiments. Nodes in a system 100 and other systems herein can create and verify digital signatures using the steps depicted for a signature creation step 338 and a signature verification step 340. In exemplary embodiments, a DPP server 103 or a responder proxy 102 could conduct a signature creation step 338 and an initiator 101 operating in a device 106 can conduct a signature verification step 340. In Figure 3d, signature creation 338 can use the sub-steps of obtaining a message to sign, calculating a message digest 338a, using a private key such as responder ephemeral private key pr 102r, using a signature algorithm 338b, inputting parameters 199i, and calculating a resulting signature 338c. The steps and sub-steps depicted for DPP server creating signature 338 in Figure 3d can also be used by other nodes, including signature creation 338d for a responder 102' to create a signature 338c, where responder 102' creates a signature 338c for network credentials 109 received from a DPP server 103 in Figure 6 below.

Signature algorithm 338a and a corresponding signature verification algorithm for a signature verification step 340 below could comprise an RSA-based digital signature algorithm (DSA) or an ECC based elliptic curve digital signature algorithm (ECDSA), and other possibilities exist as well for signature algorithm 338a and the corresponding signature verification algorithm without departing from the scope of the present disclosure. When using a DSA or ECDSA algorithm in non-deterministic mode for a signature creation 338, a value of "k" or "r", which could comprise a random number can be associated with the digital signature 338c. When using a DSA or ECDSA in deterministic mode for preferred exemplary embodiments, such as specified in IETF RFC 6979 and

titled “Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)”, which are hereby incorporated by reference, then the requirement for a separately transmitted random number with digital signature 338c (such as value “k” or “r”) can be optionally omitted, such that “r” can be deterministically calculated based on the message to sign. In
 5 exemplary embodiments, device 106 and servers such as DPP sever 103 can utilize deterministic ECDSA without also sending a random number along with a digital signature, although the value “r” from the deterministic mode could be sent with the digital signature 338c. In other words, a value can be sent with the digital signature 338c that is deterministically derived and associated with the message to sign. In other exemplary embodiments, a random number can be generated a derived value for the
 10 random number such as “r” sent with digital signature 338c (e.g. not in deterministic mode).

For a signature creation 338 step, the exemplary message to sign comprises either plaintext network credentials 109 or asymmetrically encrypted network credentials 109'. The message to sign values can be transmitted to the initiator 101 in an encrypted format, such as shown for message 124 in Figure 3b. The message to sign values can be input into a message digest algorithm 338a, which
 15 could comprise a standard algorithm such as SHA-256, SHA-3, or similar algorithms. In exemplary embodiments, the message digest algorithm 338a can be specified by a hash value 199e in a selected set of parameters 199a from parameters 199. The output of message digest algorithm 338a can be input along with parameters 199i and a private key such as pr 102d into signature algorithm 338b. Parameters 199i can specify encoding rules, padding, key lengths, selected algorithms, curve names,
 20 and other values or fields necessary to utilize a signature algorithm 338b. Both a signature creation step 338 and a signature verification step 340 use the same or equivalent values for parameters 199i. A private key used in a signature creation step 338 can comprise a private key held and recorded by a network 107 where initiator 101 can record the corresponding public key.

Figure 3d depicts the private key as responder ephemeral private key pr 102r, although
 25 alternatively the private key could be responder bootstrap private key br 102b or a different compatible private key recorded by network 107. The selection of (i) key pr 102r or key br 102b for a signature creation step 338 and (ii) key Pr 102p or key Br 102a for a signature verification step 340 can be specified in a set of cryptographic parameters 199a, or simply included in software for DPP server 103 and initiator 101. For embodiments where signature 338 uses a private key of responder bootstrap private key br 102b, then initiator 101 could record and use responder bootstrap public key Br 102a.
 30 For embodiments where signature 338 uses a private key for network 107, then initiator 101 could record a certificate for the network 107 and use the public key from the certificate for a signature verification step 340.

Signature verification 340 can comprise a step using the sub-steps of (i) obtaining a message
 35 to verify, (ii) calculating a message digest 338a for the message to verify, (iii) using a public key corresponding to the private key, such as responder ephemeral public key Pr 102p for responder

ephemeral private key pr_{102r} , (iv) using a signature verification algorithm corresponding to signature creation algorithm 338b, (v) inputting parameters 199i and received signature 338c into the signature verification algorithm, and (vi) determining a pass/fail. If the signature 338c received matches a calculated signature using the public key and message to verify, then the received signature 338c passes or is validated or verified. If the signature 338c does not match the calculated signature in a step 340, then the signature 338c is considered to fail or not be verified. In exemplary embodiments, a device 106 operating an initiator 101 can conduct a signature verification step 340 for a set of credentials 109 (or encrypted credentials 109') before accepting and applying the credentials in a configuration step 106a.

Figure 3e

Figure 3e is a flow chart illustrating exemplary steps for using asymmetric ciphering in order to encrypt credentials using a public key and decrypting credentials using the corresponding private key, in accordance with exemplary embodiments. An encryption 334 step could be performed by (i) a responder 102' as depicted and described in connection with Figure 3b above in order to create a ciphertext 334a, (ii) an external network 108 (different than network 107 operating DPP server 103), where the external network 108 could be the source of a plaintext set of credentials 109 for device 106. The purpose of an encryption 334 step can be to convert a plaintext set of network credentials 109 into the ciphertext 334a, which can comprise and be depicted as an asymmetrically encrypted set of network credentials 109' in Figure 3b, Figure 3c, Figure 6, etc. An exemplary reason that an asymmetrically encrypted set of network credentials 109' may be preferred for receipt by and operation inside DPP server 103 is that DPP server 103 is that a network access point 105 or an external network 108 may prefer that DPP server 103 or responder 102' not be able to read asymmetrically encrypted set of network credentials 109', while device 106 could decrypt and read the plaintext set of network credentials 109'.

An encryption step 334 can use an asymmetric encryption algorithm 351a, where asymmetric encryption algorithm 351a can be based on either RSA or ECC algorithms. The use of an asymmetric encryption algorithm 351a and an asymmetric decryption algorithm 351b with RSA based keys is described in Internet Engineering Task Force Request for Comments (RFC) 8017, which is titled "PKCS #1: RSA Cryptography Specifications Version 2.2", which is hereby incorporated by reference. The use of asymmetric encryption algorithm 351a and an asymmetric decryption algorithm 351b with elliptic curve cryptography can be accomplished with ElGamal asymmetric encryption, as summarized in the Wikipedia article for ElGamal encryption from April 3, 2018, which is herein incorporated by reference. Other possibilities exist as well for the use of ECC algorithms for

asymmetric encryption, including IEEE 1363a standards and ISO/IEC standard 18033-2. Other possibilities exist as well without departing from the scope of the present disclosure.

For an encryption step 334, the initiator ephemeral public key P_i 101c can be input into an asymmetric encryption algorithm 351a. As depicted, the plaintext values can be input as well, along with a set of parameters 199h. Parameters 199h can be a subset of parameters 199 depicted and described in connection with Figure 2a above, and can include a set of asymmetric encryption parameters 199h. Note that where initiator 101 or responder 102' is depicted as recording or using a set of parameters 199a, the set of parameters 199a can specify a selected row in a set of parameters 199, such that a single value for parameters 199a can identify all values for 199a through 199h in a set of cryptographic parameters 199. Parameters 199h can specify key lengths, encoding rules, byte orders, use of point compression, and other values necessary to operate an asymmetric ciphering algorithms 351a and 351b. The output of an asymmetric encryption algorithm 351a can be ciphertext 334a.

In a different, but related, exemplary embodiment, the public key used in an encryption step 334 can comprise a public key from certificate cert.device 106m as depicted for device 106 in Figure 1a. The public key from certificate cert.device 106m may be more accessible to an external network 107 or a node than key P_i 101c. The parameters 199h used for a step 334 in this different embodiment could be recorded in the certificate cert.device 106m. Cert.device 106m could be publicly shared by a device manufacturer or device distributor along with a device identity in the certificate such as ID.device 106i or MAC.I 106n. For this different, but related embodiment of encryption step 334, the secret key for device 106 or initiator 101 to utilize in a decryption step 339 below can comprise the secret key SK.device 106s.

As depicted in Figure 3e, and encryption 334 step could also comprise an embodiment of encrypting a plaintext symmetric key using asymmetric encryption, such that the symmetric key could be used with a symmetric ciphering algorithms 405a and 405b such as AES, AES-SIV, or Triple Data Encryption Standard (3DES) or Blowfish, or related algorithms. In this embodiment, (i) the symmetric key for encryption network credentials 109 could comprise a random number derived by responder 102' (or a node for the owner of credentials 109), and (ii) ciphertext 334a could comprise two parts, where the first part is the encrypted symmetric key and the second part is a symmetric encryption of network credentials 109. In other words, for this embodiment where (a) a symmetric key for encrypting/decrypting credentials 109 is asymmetrically encrypted with algorithm 351a, then (b) network credentials 109 would be symmetrically encrypted/decrypted using the asymmetrically encrypted/decrypted symmetric key.

For a decryption step 339, the initiator ephemeral private key p_i 101d for initiator 101 in device 106 can be input into an asymmetric decryption algorithm 351b. The initiator ephemeral private key p_i 101d can be the secret key corresponding to the public key P_i 101c used with the asymmetric

encryption algorithms 351a. Asymmetric decryption algorithm 351b can correspond to asymmetric encryption algorithm 351a, such that if an ECC curve with a given set of parameters 199h is used with asymmetric encryption algorithm 351a, then the same ECC curve with the same or equivalent set of parameters 199h can be used with asymmetric decryption algorithm 351b. The output of an asymmetric decryption algorithm 351b can be the plaintext, as depicted in Figure 3e.

In addition, although credentials 109 are depicted as decrypted in a decryption step 339, an asymmetric decryption algorithm 351b could be used to read a plaintext symmetric key for a symmetric ciphering algorithm, where the plaintext credentials 109 could be read by inputting the plaintext symmetric key and encrypted credentials 109 into a symmetric ciphering algorithm 405b. Standards such as IEEE 1363a standards and ISO/IEC standard 18033-2 describe using asymmetric algorithms and public keys to encrypt symmetric keys, and asymmetric decryption algorithms and private keys to decrypt the asymmetrically encrypted symmetric keys, where (i) the plaintext and ciphertext can be converted using the symmetric key, and then (ii) a second plaintext comprising credentials 109 could be read using the symmetric key and a symmetric ciphering algorithm. In other words, as described in two paragraphs above, the ciphertext 334a could comprise two parts, where (i) the first part comprises a symmetric key asymmetrically decrypted by 351b, and then (ii) the second part comprises the symmetrically encrypted credentials 109, where the symmetrically encrypted credentials 109 are decrypted with the asymmetrically decrypted (e.g. plaintext) symmetric key.

Note that encryption step 334 and decryption step 339 can be used with other plaintext values and ciphertext values as contemplated herein, such as (i) a device 106 asymmetrically encrypting in a step 334 a configuration attribute 330a for an external network 108 using a public key for the external network, and then (ii) the external network 107 asymmetrically decrypting in a step 339 the configuration attribute 330a using a private key. Other possibilities exist as well without departing from the scope of the present disclosure.

Figure 4a

Figure 4a is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key, and for using the derived shared secret key to encrypt and decrypt data, in accordance with exemplary embodiments. Exemplary steps for a responder 102' and an initiator 101 to mutually derive keys can comprise (i) a key exchange 319 step for DPP server 103 or responder 102' (depending on the responder mode 306a) to derive a symmetric encryption key k1 314a and (ii) a key exchange 314 step for a initiator 101 to derive the same symmetric encryption key k1 314a. Exemplary steps for a initiator 101 to encrypt data and responder 102' or DPP server 103 (depending on a mode 306a) to decrypt data can comprise (a) an encryption step 315 for initiator 101 to utilize the symmetric encryption key k1 314a to convert plaintext to ciphertext, and (b) a decryption

320 step for responder 102' or DPP server 103 (depending on mode 306a) to utilize the symmetric encryption key k1 314a to convert the ciphertext received from initiator 101 into plaintext.

The use of the steps for a key exchange 314, a key exchange 319, encryption 315, and decryption 320 were also depicted and described in connection with Figure 3a and Figure 3b above.

Additional detail regarding the use of these steps will be provided herein. Further, although the steps are depicted specifically for the use of particular keys and plaintext/ciphertext combinations in Figure 4a, the steps illustrated can be used with different keys and plaintext/ciphertext combinations. For example, Figure 4b below depicts the same functionality used by the nodes as in Figure 4a, but with (i) different PKI keys input to mutually derive a different symmetric encryption key, (ii) a different key derivation function and (iii) different plaintext encrypted and different ciphertext decrypted.

A key exchange 314 step for initiator 101 to derive a symmetric encryption key 314a can utilize a selected set of cryptographic parameters 199a as depicted and described in connection with Figure 1d and Figure 1e above. As depicted in Figure 4a, a key exchange algorithm 401a can receive input both of a responder bootstrap public key Br 102a and an initiator ephemeral private key pi 101d. The key exchange algorithm 401a could comprise a Diffie Hellman key exchange (DH), an Elliptic Curve Diffie Hellman key exchange (ECDH), and other possibilities exist as well without departing from the scope of the present disclosure. A key exchange algorithm 401a can support either PKI keys based on elliptic curves or RSA algorithms, although support of elliptic curves may be preferred in some exemplary embodiments due to their shorter key lengths and lower processing requirements.

A summary of ECDH as a key exchange algorithm 401a is included in the Wikipedia article titled "Elliptic Curve Diffie-Hellman" from March 9, 2018, which is herein incorporated by reference. An exemplary embodiment of key exchange algorithm 401a could comprise a "One-Pass Diffie-Hellman, C(1, 1, ECC CDH)" algorithm as described in section 6.2.2.2 on page 81 of the National Institute of Standards and Technology (NIST) document "NIST SP 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" from March, 2007 which is hereby incorporated by reference its entirety. Other key exchange algorithms in NIST SP 800-56A could be utilized as well for a key exchange algorithm 401a in Figure 4a, Figure 4b, and Figure 4c without departing from the scope of the present disclosure. Example calculations for an ECDH key exchange for a key exchange algorithm 401a are shown on pages 88 and 89 of DPP specification version 1.0, where a selected set of parameters 199a can use the ECC curve P-256 with PKI key lengths of 256 bits.

Other algorithms to derive a shared symmetric encryption key using public keys and private keys may also be utilized in a key exchange algorithm 401a (and 401b in Figure 4b and 401c in Figure 4c), such as, but not limited to, the American National Standards Institute (ANSI) standard X-9.63. Cryptographic parameters 199a can also include information for using a key exchange algorithm 401a and a key derivation function 402a in order to derive a commonly shared symmetric encryption key

k1 314a. As contemplated herein, the terms “selected set of cryptographic parameters 199a” and “cryptographic parameters 199a”, and “parameters 199a” can be equivalent. Parameters 199a input into a key exchange algorithm 401a can include a time-to-live for a key k1 314a that is derived, a supported point formats extension, where the supported point formats extension could comprise uncompressed, compressed prime, or “compressed char2” formats, as specified in ANSI X-9.62. In other words, an ECC keys input into a key exchange algorithm 401a may have several different formats and a set of parameters 199a can be useful to specify the format.

As depicted in Figure 4a, the output of a key exchange algorithm 401a, such as an ECDH key exchange, can comprise a secret shared key M 410. The secret shared key M 410 can be input into both (i) a first key derivation function 402a in order to calculate the key k1 314a and (ii) a second key derivation function 402c in order to calculate a symmetric encryption key ke 327a as depicted in Figure 4c below. Key derivation function 402a could use a hash algorithm 199e from parameters 199a, where an input into the hash algorithm 199e can include the value M 410, in addition to other data such as public keys. As contemplated herein, a secret shared key can also be referred to as a value, since the secret shared key can be a numeric value or a number in the form of a sequence of digits. In DPP specification version 1.0 a key derivation function to determine key k1 314a is depicted on the fourth line of page 48.

Note that in some exemplary embodiments, such a second responder mode 306a depicted and described in connection with Figure 5a below, (i) a key exchange algorithm 401a can be processed by a DPP server 103 and a key derivation function 402a can be processed by an responder 102'. In other words, for some embodiments the key exchange 314b step can be shared or distributed between a DPP server 103 and an responder 102'. Other possibilities exist as well for a key derivation function 402a without departing from the scope of the present disclosure. For example, with other embodiments besides those within DPP specification version 1.0, the output of a key derivation function 402a can be both a symmetric encryption key k1 314a and a message authentication code (MAC) key, where a MAC key can function to verify message integrity of ciphertexts generated with the symmetric encryption key 314a.

As depicted in Figure 4a, an encryption 315 step could be utilized with the derived symmetric encryption key k1 314a in order to convert plaintext 403a into ciphertext 315a. An encryption 315 step can utilize a symmetric encryption algorithm 405a, where symmetric encryption algorithm 405a could comprise an algorithm according to the Advanced Encryption Standard (AES), Blowfish, and Triple Data Encryption Standard (3DES), and other possibilities exist as well. As depicted in Figure 4a, a symmetric encryption algorithm 405a can comprise the AES-SIV algorithm as specified in IETF RFC 5297. Parameters 199f could specify settings for a symmetric ciphering algorithm 405a as well, such as key length and a block cipher modes of ECB, CBC, OFB, and CFB. Parameters 199f may

also specify an authenticated encryption mode such as CCM (NIST SP800-38C), GCM (NIST SP800-38D), and other examples exist as well.

Symmetric encryption algorithm 405a can accept input of (i) plaintext 403a, (ii) symmetric encryption key k1 314a, (iii) parameters 199f, and (iv) optionally an initialization vector (not shown in Fig. 4a and not required for AES-SIV), and output ciphertext 315a. Although not depicted in Figure 4a, and encryption 405a step can output a message authentication code (MAC), where the MAC can be calculated with the MAC key from a key derivation function 402a. In exemplary embodiments the use of a MAC and MAC key, as well as an initialization vector can be optionally omitted, such as with algorithm 405a. The exemplary listed values for plaintext and ciphertext in step 315 in Figure 4a were described in connection with depicted step 315 in Figure 3a.

Key exchange 319 step for a responder 102' or DPP server 103 (depending on a mode 306a) depicted in Figure 4a can correspond to key exchange 319 in Figure 3a. Key exchange 319 step can comprise inputting or using the initiator ephemeral Public key Pi 101c (from message 122) and the responder bootstrap private key br 102b into key exchange algorithm 401a, which can comprise the same or equivalent key exchange algorithm 401a depicted and described in connection with key exchange step 314 described above. Other elements or algorithms within a key exchange step 319 can be equivalent to a key exchange step 314 above, including the use of shared parameters 199a. In this manner, responder 102' or DPP server 103 (depending on a mode 306a) and initiator 101 can securely derive the same symmetric encryption key k1 314a as depicted in Figure 4a.

For embodiments with a first mode 306a depicted in Figure 3a, DPP server 103 can conduct key exchange step 319 and for embodiments with a second mode 306a depicted in Figure 5a, a responder 102' can conduct key derivation function 402a and DPP server 103 can conduct key exchange algorithm 401a. For embodiments where DPP server conducts a key exchange step, such as step 401a in Figure 4a, or other key exchange steps such as 401a in Figure 4b below, etc., the appropriate keys for input into the key exchange algorithm can be selected from a database (e.g. 103a for DPP server) or table (e.g. table 102u) using an identity for device 106 received in message 122. In exemplary embodiments, the identity of device 106 can comprise either MAC.I 106n or the hash value 101aa over the initiator bootstrap public key Bi 101a.

A decryption 320 step allows responder 102' or DPP server 103 (depending on mode 306a) to convert the ciphertext 315a received in a message 122 from Figure 3a into plaintext 403a. Decryption 320 step can utilize a symmetric decryption algorithm 405b, which could comprise the same algorithm used in symmetric encryption algorithm 405a except the algorithm being used for decryption instead of encryption. For example, if symmetric encryption algorithm 405a in Figure 4a comprises an AES-SIV encryption, then symmetric decryption algorithm 405b can comprise an AES-SIV decryption. Note that the same values are input into symmetric decryption algorithm 405b as symmetric encryption algorithm 405a, such as symmetric encryption key k1 314a and parameters 199f in order to convert

ciphertext 315a back into plaintext 403a. Responder 102' or DPP server 103 (depending on the mode 306a) can read and process plaintext 403a after a decryption 320 step. For example, with the first mode 306a depicted in Figure 3a and Figure 3b, DPP server 103 can perform decryption step 320 to read plaintext 403a. With the second mode 306a depicted in Figure 5, responder 102' can perform decryption step 320 to read plaintext 403a.

For embodiments where a MAC code is received with ciphertext 315a, initiator 101 can also verify the MAC code using a MAC key from a key derivation function 402a in order to verify the integrity of ciphertext 315a. Note that AES-SIV can have a message integrity check built into the algorithm, and thus a separate MAC code and MAC key can be optionally omitted in some exemplary embodiments. In exemplary embodiments, the successful decryption of a ciphertext into a plaintext using decryption algorithms 405b supports mutual authentication between the two nodes, since successful decryption can only take place when the two nodes record the depicted private keys used for the depicted key exchange.

Figure 4b

Figure 4b is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key, and for using the derived shared secret key to encrypt and decrypt data, in accordance with exemplary embodiments. Exemplary steps for (i) an responder 102' or a DPP server 103 (depending on the mode 306a) and (ii) an initiator 101 to mutually derive keys can comprise (a) a key exchange 314b step for DPP server 103 or responder 102' (depending on the mode 306a) to derive a symmetric encryption key k2 314c and (b) a key exchange 319a step for an initiator 101 to derive the same symmetric encryption key k2 314c. Exemplary steps for an initiator 101 to decrypt data and responder 102' (or DPP server 103) to encrypt data can comprise (i) an encryption 315b step for responder 102' or DPP server 103 to utilize the symmetric encryption key k2 314c to convert plaintext to ciphertext, and (ii) a decryption 320b step for initiator 101 to utilize the symmetric encryption key k2 314c to convert the ciphertext received from responder 102' into plaintext. The use of the steps for a key exchange 314a, a key exchange 319a, encryption 315b, and decryption 320b were also depicted and described in connection with Figure 3a above.

Note that steps in Figure 4b and the steps in Figure 4a can share some algorithms and values, and the descriptions for the algorithms and values in Figure 4a can be applicable for Figure 4b. For example, a same key exchange algorithm 401a depicted and described in Figure 4a can also be used in Figure 4b. The key exchange algorithm 401a can comprise an ECDH key exchange. The set of parameters 199a depicted and described in Figure 4a can also be used in Figure 4b. Further, a symmetric encryption algorithm 405a and a symmetric decryption algorithm 405b can be the same in

Figure 4a and Figure 4b. A set of parameters 199f for a symmetric encryption algorithm 405a and a symmetric decryption algorithm 405b can be the same in Figure 4a and Figure 4b.

A responder 102' or DPP server 103 (depending on a mode 306a) can conduct a key exchange step 314b. In a key exchange step 314b, the initiator ephemeral public key P_i 101c received in message 122 from Figure 3a can be input into an ECDH key exchange algorithm 401a along with the responder ephemeral private key pr 102r derived in a step 318 in Figure 3a in order to output a secret shared key N 411. The value for shared secret key N 411 can be recorded and also used in the derivation of a key ke 327a in a key exchange step 321 below in Figure 4c. For a step 314b, the value for the shared secret key N 411 can be input into a key derivation function 402b in order to derive a mutually shared symmetric key k_2 314c. In exemplary embodiments which support DPP specification version 1.0, a key derivation function 402b can comprise the key derivation function for key k_2 in point 2 of section 6.2.3 of DPP specification version 1.0. For a first responder mode 306a depicted in Figure 3a, a DPP server 103 can conduct key exchange step 314b and for a second responder mode 306a depicted in Figure 5a, a responder 102' can conduct key exchange step 314b.

An initiator 101 can conduct a key exchange step 319a. At step 319a, a received, validated responder ephemeral public key Pr 102p and a derived initiator ephemeral private key pi 101d can be input into an ECDH key exchange algorithm 401a in order to calculate a shared secret key N 411. Shared secret key N 411 can also be recorded and used in a step 327 below. Initiator 101 can input shared secret key N 411 into a key derivation function 402b in order to mutually derived the symmetric encryption key k_2 314c. In exemplary embodiments which support DPP specification version 1.0, a key derivation function 402b can comprise the key derivation function for key k_2 in section 6.2.4 of DPP specification version 1.0.

Responder 102' or DPP server 103 (depending on a mode 306a) can conduct an encryption step 315b. The use for an encryption step 315b for a DPP server 103 in a first responder mode 306a is depicted and described in connection with Figure 3a above. The use for an encryption step 315b for responder 102' for a second responder mode 306a is depicted and described in connection with Figure 5a below. Ciphertext 324a can comprise a ciphertext from an encryption step 322 below in Figure 4c. In other words, a responder 102' or DPP server 103 can conduct the encryption step 322 below and also key exchange step 321 below in Figure 4c before conducting the encryption step 315b. As depicted in Figure 4b, the ciphertext 324a can also be encrypted again for an encryption step 315b. The symmetric ciphering key for encryption step 315b can comprise symmetric key k_2 314c from a key exchange step 314b in two paragraphs above. The output from an encryption step 315b can be ciphertext 315c, as depicted in Figure 4b. The plaintext 403b for an encryption step 315b can comprise the depicted values of ciphertext 324a, I-nonce 317a, R-Capabilities 318c, and R-nonce 318b.

A decryption 320b step can be performed by an initiator 101. A decryption 320b step converts the ciphertext 315c received in a message 123 from Figure 3a into plaintext 403b. Decryption 320b

step can utilize a symmetric decryption algorithm 405b, which could comprise the same algorithm used in symmetric encryption algorithm 405a except the algorithm being used for decryption instead of encryption. Note that the same values (other than ciphertext) are input into symmetric decryption algorithm 405b as symmetric encryption algorithm 405a, such as symmetric encryption key k2 314c and parameters 199f in order to convert ciphertext 315c back into plaintext 403b. Initiator 101 can read and process plaintext 403b after a decryption 320b step. In exemplary embodiments, the successful decryption of a ciphertext into a plaintext using decryption algorithms 405b supports mutual authentication between the two nodes, since successful decryption can only take place when the two nodes record the depicted private keys used for the depicted key exchange.

Figure 4c

Figure 4c is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key, and for using the derived shared secret key to encrypt and decrypt data, in accordance with exemplary embodiments. Exemplary steps for (i) an responder 102' or a DPP server 103 (depending on the mode 306a) and (ii) an initiator 101 to mutually derive keys can comprise (a) a key exchange 321 step for DPP server 103 or responder 102' (depending on the mode 306a) to derive a symmetric encryption key ke 327a and (b) a key exchange 327 step for an initiator 101 to derive the same symmetric encryption key ke 327a. Note that in some exemplary embodiments, such a second responder mode 306a depicted and described in connection with Figure 5a below, (i) a key exchange algorithm 401c can be processed by a DPP server 103 a key derivation function 402c can be processed by an responder 102'. In other words, for some embodiments the key exchange 321 step can be shared or distributed between a DPP server 103 and an responder 102'.

Exemplary steps in Figure 4c for (i) a responder 102' or DPP server 103 to encrypt data and initiator 101 to decrypt data can comprise (i) an encryption 322 step for responder 102' or DPP server 103 to utilize the symmetric encryption key ke 327a to convert plaintext to ciphertext, and (ii) a decryption 328 step for initiator 101' to utilize the symmetric encryption key ke 327a to convert the ciphertext received from responder 102' into plaintext. The use of the steps for a key exchange 321, a key exchange 327, encryption 322, and decryption 328 were also depicted and described in connection with Figure 3a and Figure 3b above.

Note that steps in Figure 4c and the steps in Figure 4a can share some algorithms and values, and the descriptions for the algorithms and values in Figure 4a can be applicable for Figure 4c. For example, the key exchange algorithm 401c can comprise an ECDH key exchange. The set of parameters 199a depicted and described in Figure 4a can also be used in Figure 4c. Further, a symmetric encryption algorithm 405a and a symmetric decryption algorithm 405b can be the same in

Figure 4a and Figure 4c. A set of parameters 199f for a symmetric encryption algorithm 405a and a symmetric decryption algorithm 405b can be the same in Figure 4a and Figure 4c.

For a first responder mode 306a depicted in Figure 3a and Figure 3b, a DPP server 103 can conduct a key exchange step 321. For a second responder mode 306a depicted in Figure 5a, a key exchange step 321 can be shared between DPP server 103 and responder 102', such that DPP server 103 conducts a key exchange algorithm 401c and responder 102' conducts a key derivation function 402c. In a key exchange step 321, (i) the initiator ephemeral public key P_i 101c received in message 122 from Figure 3a can be input into an ECDH key exchange algorithm 401c along with (ii) the responder ephemeral private key pr 102r and the responder bootstrap private key br 102b.

An ECDH key exchange algorithm 401c can use $(pr\ 102r + br\ 102b)$ modulo q , as specified in DPP specification version 1.0 in point 2 of section 6.2.3, so algorithm 401c for initiator 101 can be slightly different than algorithm 401a for initiator 101 above in Figure 4a and Figure 4b. The output of key exchange algorithm 401c can be a shared secret key L 412. The value L 412 can be input into a key derivation function 402c in order to calculate a shared secret key ke 327a. Note that for a key derivation function 402c, the input can include all of key M 410 from a step 319, key N from a step 314b, and the derived key L 412. As contemplated herein, a key derivation function 402a, 402b, and 402c can also use other data mutually shared between (i) initiator 101 and (ii) responder 102' or DPP server 103 (depending on a mode 306a). In exemplary embodiments which support DPP specification version 1.0, a key derivation function 402c can comprise the key derivation function for key ke 327a in point 2 of section 6.2.3 of DPP specification version 1.0.

An initiator 101 can conduct a key exchange step 327. At step 327, (i) a combination of a recorded responder bootstrap public key Br 102a and received responder ephemeral public key Pr 102p, and (ii) the recorded initiator bootstrap private key bi 101b can be input into an ECDH key exchange algorithm 401c in order to calculate a shared secret key L 412. The recorded responder bootstrap public key Br 102a and received responder ephemeral public key Pr 102p can be combined via elliptic curve point addition. Shared secret key L 412 can be input into a key derivation function 402c, along with shared secret key M 410 from a step 314 and shared secret key N 411 from a step 319a in order to mutually derive the symmetric encryption key ke 327a. In exemplary embodiments which support DPP specification version 1.0, a key derivation function 402c can comprise the key derivation function for key ke 327e in section 6.2.4 of DPP specification version 1.0.

Responder 102' or DPP server 103 (depending on a mode 306a) can conduct an encryption step 322, where the use for an encryption step 322 are depicted and described in connection with Figure 3a above and also Figure 4b above. Plaintext 403c in a step 322 can comprise the responder authentication value 404, where value R -auth 404 can comprise a hash 199e that includes at least the initiator nonce 317a, the second random number R -nonce 318b from a step 318 above in Figure 3a, and public PKI keys such as responder bootstrap public key 102a and responder ephemeral public key

102c. The value for R-auth 404 in a step 322 can comprise the value in point 3 of section 6.2.3 of DPP specification version 1.0. The symmetric ciphering key for encryption step 322 can comprise symmetric key ke 327a from a key exchange step 321.

5 The output from an encryption step 322 can be ciphertext 324a, as depicted in Figure 4c. Note that ciphertext 324a in a step 322 can also be encrypted again in a step 315b above in Figure 4b. In other words, a responder 102' or DPP server 103 (depending on the mode 306a) can conduct the encryption step 322 herein for Figure 4c (along with the key exchange 321 herein in Figure 4b) before conducting the encryption step 315b. For a first mode 306a representing an embodiment depicted in Figure 3a, DPP server 103 can conduct encryption step 322. For a second mode 306a representing an
10 embodiment depicted in Figure 5a, a responder 102' can conduct encryption step 322.

A decryption step 328 can be performed by initiator 101. A decryption 328 step converts the ciphertext 324a received in a message 123 from Figure 3a into plaintext 403c. Decryption 328 step can utilize a symmetric decryption algorithm 405b, which could comprise the same algorithm used in symmetric encryption algorithm 405a except the algorithm being used for decryption instead of
15 encryption. Note that the same values are input into symmetric decryption algorithm 405b as symmetric encryption algorithm 405a in Figure 4c, such as symmetric encryption key ke 327a and parameters 199f in order to convert ciphertext 324a back into plaintext 403c. Initiator 101 or device 106 can then read and process plaintext 403c after a decryption 328 step. In exemplary embodiments, the successful decryption of a ciphertext into a plaintext using decryption algorithms 405b supports
20 mutual authentication between the two nodes, since successful decryption can only take place when the two nodes record the depicted private keys used for the depicted key exchange. In other words, only the nodes could derive key ke 327a in Figure 4c, and thus data that is successfully encrypted by one node and decrypted by the other node using key ke 327a would confirm the nodes are authenticated, since only the nodes recording the private keys depicted in Figure 4c would be able to
25 derive keys ke 327a.

As depicted and described in connection with Figure 3a, a responder 102' or DPP server 103 can also conduct both an encryption step 322 and a decryption step 328. The steps for DPP server 103 to conduct a decryption step 328 for a first responder mode 306a can comprise step 328b and step 328c as depicted and described in Figure 3a. When responder 102' or DPP server 103 conducts
30 decryption step 328 using symmetric encryption key ke 327a, the ciphertext and plaintext will comprise different values than those depicted in Figure 4c. For example, in Figure 3a a DPP server 103 with a first responder mode 306a can use a decryption step 328b (e.g. step 328 but with different ciphertext than that depicted in Figure 4c) in order to decrypt the encrypted initiator authentication value 404a into a plaintext authentication value 404a. Also, in Figure 5a, a responder 102' for a second
35 responder mode 306a can use a decryption step 328b (e.g. step 328 but with different ciphertext 324e than that depicted in Figure 4c) in order to decrypt the encrypted initiator authentication value 404a.

Further, an initiator 101 can use a decryption step 328d to decrypt network credentials 109 from a responder 102'.

Note that in some exemplary embodiments, the "plaintext" network credentials 109' (for the purposes of a step 328d in Figure 5a) can comprise network credentials 109' that can also be encrypted with an asymmetric ciphering algorithm such as Elgamal using a step 351a in Figure 3e above. In other words, network credentials 109' can comprise credentials that are "double encrypted" with both (i) a symmetric key 327a (which can be decrypted by initiator 101 using a decryption step 328d, where decryption step 328d uses the depicted decryption 328 in Figure 4c), and then (ii) an asymmetric ciphering algorithm 351b such as Elgamal from a step 339. In other words, the output of a decryption step 328d could be input into an asymmetric decryption algorithm 351b, so that the input into asymmetric decryption algorithm 351b could be decrypted by initiator 101 using either (i) initiator ephemeral private key pi 101d or (ii) initiator bootstrap private key bi 102b.

Figure 4d

Figure 4d is a flow chart illustrating exemplary steps for conducting key exchanges using PKI keys in order to derive a first and a second shared secret keys, where ECC point addition is performed on output from a key exchange algorithm in order to derive a third shared secret key. The exemplary key exchange step 421 can be used by a DPP server 103 in order to derive a key L(a) 421a. The exemplary key exchange step 422 can be used by a responder 102' in order to derive a key L(b) 422a. A key derivation step 423 can combine key L(a) 421a and key L(b) 422b in order to calculate or derive the key L 412, using ECC point addition on key L(a) 421a and key L(b) 422b. In this manner, key L 412 can be calculated through the use of two different key exchange steps 421 and 422 in some exemplary embodiments such as those depicted and described in connection with Figure 5a below, instead of the single key exchange step 321 depicted in Figure 4c.

Key exchange 421 can use an ECDH key exchange algorithm 401a, where the input comprises the initiator bootstrap public key Bi 101a and the responder bootstrap private key br 102b. The set of mutually agreed parameters 199a between with an initiator 101 can be used with a key exchange step 421. The output of ECDH key exchange algorithm 401a can comprise key L(a) 421a. In exemplary embodiments, where DPP server 103 conducts the key exchange step 421, the responder bootstrap private key br 102b can remain in DPP server 103 or within a network 107 and does not need to be shared with responder 102'. In addition, although a DPP server 103 is described herein for a Figure 4d as performing the key exchange step 421, another server on an external network 108 could (i) record the responder bootstrap private key br 102b and (ii) perform a key exchange step 421, and the output of key exchange step 421 in the form of key L(a) 421a could be sent to a DPP server 103 or

responder 102', such as in a message 503 depicted in Figure 5a below. Or, DPP server 103 could be operated by an external network 108 and communicate with a responder 102' via a secure session 302b. Key L(a) 421a could comprise a shared secret key, although in exemplary embodiments an initiator 101 may not need to calculate the corresponding key L(a) 421a directly, but rather initiator 101 can calculate a shared secret key L 412 using step 327 as depicted in Figure 4c.

Key exchange 422 can use an ECDH key exchange algorithm 401a, where the input comprises the initiator bootstrap public key Bi 101a and the responder ephemeral private key pr 102r. The set of mutually agreed parameters 199a between with an initiator 101 can be used with a key exchange step 422. The output of ECDH key exchange algorithm 401a can comprise key L(b) 422a. In exemplary embodiments, where responder 102' conducts the key exchange step 422, the responder does need access to the responder bootstrap private key br 102b, which can remain in DPP server 103 or within a network 107 and does not need to be shared with responder 102'. Key L(b) 422a could comprise a shared secret key, although in exemplary embodiments an initiator 101 may not need to calculate the corresponding key L(b) 421a directly, but rather initiator 101 can calculate a shared secret key L 412 using a step 327 as depicted in Figure 4c.

A key derivation step 423 can (i) combine the output of key exchange steps 421 and 422 in order to calculate or derived the shared secret key L 412 and then (ii) perform a key derivation function step 402c on the derived or calculated shared secret key L 412 in order to determine or calculate shared secret key ke 327a, which can be also mutually derived by initiator 101, where initiator 101 uses the key exchange step 327 depicted and described in connection with Figure 4c. In exemplary embodiments, a responder 102' can conduct the key derivation step 423 using (i) the value L(a) 421a received from a DPP server 103 (where receipt of L(a) 421a by responder 102' can be in a message 503 as shown in Figure 5a below), and (ii) the value or key L(b) 422b output from a key exchange step 422 in the paragraph above.

Key derivation step 423 can comprise two primary steps. A first step in key derivation 423 can comprise an ECC point addition 424 on the value L(a) 421a and the value L(b) 422a. The result of the ECC point addition will be equal to the value L 412, where the value L 412 could also be calculated without ECC point addition 424 using a step 321. Exemplary calculations for an ECC point addition 424 can comprise the calculations shown for point addition in the Wikipedia article for "Elliptic Curve Point Multiplication" dated May 15, 2018, which is herein incorporated by reference in its entirety. As depicted in Figure 4c, the calculation of L 412 using an ECC point addition 424 over L(a) 421a and L(b) 422a will equal the value for L 412 calculated by a key exchange algorithm 401c from a step 321 in Figure 4c. A second step in key derivation step 423 as depicted in Figure 4d can comprise a key derivation function step 402c using input from ECC point addition step 424 (e.g. value L 412 output from step 424), where the output of key derivation function step 402c can comprise key ke 327a, which is also depicted in Figure 4c above.

By conducting a key derivation step 423 as depicted in Figure 4c (where DPP server 103 conducts the calculations using the responder bootstrap private key br 102b), (i) responder 102' can calculate key ke 327a without recording or operating on the responder bootstrap private key br 102b, and (ii) the responder bootstrap public key Br 102a could comprise a shared key 102z as depicted in Figure 2a across multiple different devices 106. Although the responder 102' can receive the key or value L(a) 421a, the responder bootstrap private key br 102b remains secure, because responder 102' cannot feasibly calculate or determine the responder bootstrap private key br 102b using a received key L(a) 421a. In other words, the use of a key derivation step 423 can allow a responder bootstrap public key Br 102a to be securely shared between multiple devices 106 in order to conduct a DPP 191, since a responder 102' can conduct a DPP 191 as contemplated herein without recording, receiving, or operating on a responder bootstrap private key 102b.

Exemplary data and numbers can be provided to demonstrate the calculations for key exchange steps 421, key exchange step 422, and key derivation step 423 using an ECC point addition 424. The exemplary data can comprise decimal numbers for the example data and calculations listed in "Test vectors for DPP Authentication using P-256 for mutual authentication" on pages 88 and 89 of the DPP specification version 1.0. Parameters 199a can comprise the elliptic curve 199c of "secp256r1" with key lengths 199b of 256 bit long keys.

The responder bootstrap private key 102b can comprise the exemplary following number, and may be recorded in a DPP server 103:

38358416135251014160802731750427376395128366423455574545250035236739593908128

The responder ephemeral private key 102r can comprise the exemplary following number, and may be recorded by a responder 102':

111991471310604289774359152687306247761778388605764559848869154712980108827301

The initiator bootstrap public key Bi 101a can comprise the following exemplary values with X and Y numbers (or "coordinates") of:

X:

61831688504923817367484272103056848457721601106987911548515219119661140991966

Y:

436821274116052626307636850969789027573720854595612820926922498255090826944

Key exchange step 421 for an ECDH algorithm key exchange 401a can input the initiator bootstrap public key Bi 101a and the responder bootstrap private key br 102b (both with numbers above) in order to calculate a key L(a) 421a. An exemplary number or value for key L(a) 421a from the values above using parameters 199a can be:

X:
11490047198680522515311590962599671482029417064351337303313906642805743573119
Y:
27933966560238204731245097943399084523809481833434754409723604970366082021855

Key exchange step 422 for an ECDH algorithm key exchange 401a can input the initiator bootstrap public key Bi 101a and the responder ephemeral private key pr 102r (both with numbers above) in order to calculate a key L(b) 422a. An exemplary number or value for key L(b) 422a from the values above using parameters 199a can be:

X:
78944719651774206698250588701582570633503182903415394243006529481189158194650
Y:
11227712702924684581834935828837489140201820424536062912051086382324589445237

An ECC point addition 424 for the above two derived points L(a) 421a (from keys Bi 101a and br 102b) and L(b) 422b (from keys Bi 101a and pr 102r) will result in the following point that also equals L 412.

X:
113734500629065545557893524064610113740858966831672649615565042035695230713090
Y:
68961429691307429166796760881095689348088875771334970644593306388375741965262

Note that key L 412 can also be derived by initiator 101 from a key exchange step 327 using ECDH key exchange algorithm 401c. ECDH key exchange algorithm 401c can comprise initiator 101 calculating L from keys Br 102b, Pr 102p, and bi 101b, represented by $(Br\ 102b + Pr\ 102p) * bi\ 101b$. Using hexadecimal representations of the above keys Br 102b, Pr 102p, and bi 101b from the same keys values in DPP specification version 1.0 and exemplary numbers above, the X,Y numbers or point or “coordinates” derived by initiator 101 with algorithm 401c for shared secret L on page 89 equals:

X: fb737234 c973cc3a 36e64e51 70a32f12 089d198c 73c2fd85 a53d0b28 2530fd02

Y: 9876c937 b642cce8 f604bef0 c24ce0da d2c27867 08a7575a f940a58c fe8ca3ce

The hexadecimal representation of shared secret L using keys Br 102b, Pr 102p, and bi 101b by initiator 101 equals the decimal representation of shared secret key L 412 from the ECC point addition 424 by a responder 102', where ECC point addition 424 used as input key L(a) 421a and key L(b) 422a.

After an ECC point addition 424, a key derivation step 423 in Figure 4c can input the shared secret key L 412 output from the ECC point addition 424 into a key derivation function 402c. The key derivation function 402c can comprise the same key derivation function 402c as depicted and described in connection with Figure 4c above. The key derivation function 402c can also use input of shared secret key M 410 from Figure 4a in a step 319 and shared secret key N 411 from a step 314b. The output of the key derivation function 402c can be shared secret key ke 327a. An exemplary key derivation function 402c could comprise the key derivation function in point 2 of page 48 of DPP specification version 1.0. Note that initiator 101 can mutually derive the same value or number or "coordinate" for shared secret key ke 327e using a key exchange step 321. As contemplated herein, the use of the words or description "shared secret key" can also mean "shared secret". Thus, "shared secret key M 410" can be the same as "shared secret M 410", "shared secret key N 411" can be the same as "shared secret N 411", etc.

Figure 5a

Figure 5a is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device database, a DPP server, a responder proxy, and an initiator, in accordance with exemplary embodiments. System 500a can include a device database 104, DPP server 103, responder proxy 102', and initiator 101 operating in device 106. Before starting steps and message flows depicted in Figure 5a, the initiator, DPP server, and device database depicted for system 500a in Figure 5a may previously complete exemplary message flows and steps depicted for a step 301 in Figure 3a above. Responder 102' can use data received from initiator 101 in a message 122 from step 301 (such as initiator ephemeral public key Pi 101c) in order to conduct the steps and process the messages depicted for a responder 102' in Figure 5a.

Responder proxy 102' can communicate with DPP server 103 via IP network 113, also as depicted in Figure 3a above. As contemplated herein, a "responder proxy 102'" depicted in Figure 5a may also be referred to as responder 102', and a responder 102' and responder proxy 102' may be equivalent. A difference between system 500a in Figure 5a and system 300 in Figure 3a can be that (i) responder ephemeral private key pr 102r, with corresponding (ii) responder ephemeral public key

Pr 102p are derived and recorded by a responder 102' instead of a DPP server 103. In exemplary embodiments depicted in Figure 5a the responder bootstrap private key br 102b can remain recorded in a DPP server 103 and not exposed to responder 102'. In exemplary embodiments depicted in Figure 5a the responder ephemeral private key pr 102r can remain recorded in responder 102' and not exposed to DPP server 103. The exemplary embodiment depicted in Figure 5a can comprise a second mode 306a of operation for responder 102' and DPP server 103, as depicted and described in connection with Figure 3c above.

At step 502, DPP server 103 can select a mode 306a for both responder 102' and DPP server 103. Note that the mode 306a selected is not normally observable to an initiator 101, since the full and complete set of DPP 191 messages between initiator 101 and responder 102' can be transmitted and received between the two nodes. In other words, (a) the different distribution of PKI and shared secret keys in a server database 103a and responder 102' for a first mode 306a in Figure 3a and a second mode 306a in Figure 5a (b) would not normally be observable to an initiator 101. The recording of PKI and shared secret keys in a server database is depicted in a server database 103a in Figure 3c, where the value "--" can mean that the key is recorded and/or operated by responder 102'. Step 502 can also comprise DPP server 103 selecting a responder bootstrap public key Br 102a and a responder bootstrap private key br 102b and initiator bootstrap public key Bi 101a for device 106 using MAC.I 106n, although a different value in server database 103a could be used as well, such as device identity ID.device 106i or hash value 101aa. Selecting keys Br 102a and br 102b in a step 502 can use the parameters 199a received in a message 312a in Figure 3a, in order to select keys that are compatible with received initiator ephemeral public key Pi 101c from a message 122 in Figure 3a. Also, although a step 502 is depicted in Figure 5a, a step 502 could be used in other Figures and embodiments herein, include embodiments depicted in Figure 3a and Figure 5b.

DPP server 103 can then conduct an ECDH key exchange step 401a using as input (i) the responder bootstrap private key br 102b and (ii) the initiator ephemeral public key Pi 101c received in a message 122 from Figure 3a during a step 301. This ECDH key exchange step 401a is depicted and described in connection with a key exchange step 319 depicted and described in connection with Figure 4a. The output of step 401a can be shared secret M 410. DPP server 103 can send (i) shared secret M 410 along with a device identity such as MAC.I 106n (or ID.device 106i or hash value 101aa) to (ii) responder 102' in a message 503. The use of a device identity in message 503 can be helpful for embodiments where responder 102' communicates with a plurality of devices 106 either simultaneously or in sequence.

Responder 102' can receive message 503 with shared secret M 410 and conduct a series of steps in order to prepare a DPP authentication response message 123. A step 318 in Figure 5a can comprise the same steps 318 as performed by DPP server 103 in Figure 3a, except responder 102' performs the actions in Figure 5a instead of DPP server 103 performing the actions. In summary for

a responder 102' in Figure 5a, responder 102' can derive two random numbers using a random number generator. The first random number can be used as a responder ephemeral private key pr_{102r} and the second random number can be used as a responder nonce 318b. Parameters 199a can specify the format and length of key pr_{102p} and nonce 318b in a step 318 for responder 102. Responder 102' can also derive a responder ephemeral public key Pr_{102p} using the derived key pr_{102r} . DPP server 103 can also determine responder capabilities 319b, which could comprise the capabilities for a configurator.

Responder 102' can process a key derivation function 402a using the received shared secret M 410 from a message 503. The key derivation function 402a can be equivalent to a function 402a depicted and described in connection with Figure 4a. Shared secret M 410 can be input into key derivation function 402a, and the output can be shared secret key $k1_{314a}$. Responder 102' can then perform a decryption step 320 using the derived key $k1_{314a}$, where a decryption step 320 is depicted and described in connection with Figure 4a. Decryption step 320 can be performed by responder 102' on the ciphertext 315a received in a DPP authentication request message 122. After a decryption step 320, responder 102' can read plaintext 403a with the values of the initiator nonce 317a and the initiator capabilities. For the embodiment depicted in Figure 5a, the initiator 101 can have the capabilities of an enrollee, although other possibilities exist as well without departing from the scope of the present disclosure.

Responder 102' can then conduct a second responder key exchange step 314b, where the details for a key exchange step 314b are depicted and described in connection with Figure 4b above. In summary, the initiator ephemeral public key Pi_{101c} received in message 122 can be input into an ECDH key exchange algorithm 401a along with the responder ephemeral private key pr_{102r} derived in a step 318 above in order to output a secret shared key N 411. The value for shared secret key N 411 can be recorded and used in the derivation of a key ke_{327a} in a key derivation function 402c below by responder 102'. For a step 314b, the value for the shared secret key N 411 can be input into a key derivation function 402b in order to derive a mutually shared symmetric key $k2_{314c}$. As depicted in Figure 5a, the collection of steps and messages beginning with a step 502 by DPP server 103 through key exchange step 314b can comprise a step 501, where a step 501 can also be used by a DPP server 103 and responder 102' in Figure 5b below.

In Figure 5a, a DPP server 103 can then conduct or calculate an ECDH key exchange step 421, where key exchange step 421 was depicted and described in connection with Figure 4d above. In summary, the initiator bootstrap public key Bi_{101a} and the responder bootstrap private key br_{102b} can be input into an ECDH key exchange algorithm 401d using a set of parameters 199a. The output of key exchange step 421 can comprise a value $L(a)_{421a}$. Note that step 421 can be performed before a step 501 or concurrently with a step 501. DPP server 103 can then send responder 102' a message 504, where message 504 include an identity for device 106 such as $MAC.I_{106n}$, the initiator bootstrap

public key Bi 101a, and the calculated value L(a) 421a from a key exchange step 421. Message 504 can be sent via secure session 302b established during a step 301 from Figure 3a.

Responder 102' can receive message 504 and begin performing a series of steps in order to process the data and prepare a DPP authentication response message 123. Responder 102' can conduct a key exchange step 422 in order to calculate a value L(b) 422a. In summary, the responder ephemeral private key pr 102r can be input into an ECDH key exchange algorithm 401a along with the initiator bootstrap public key Bi 101a, using a set of parameters 199a in order to calculate the value or key L(b) 422a using a key exchange step 422. Key Bi 101a can be received by responder 102' in message 504. A key exchange step 422 for responder 102' was depicted and described in connection with Figure 4d above. Responder 102' can then conduct a key derivation step 423 from Figure 4d above in order to determine or derived a secret shared key ke 327a.

A key derivation step 423 for responder 102' was depicted and described in connection with Figure 4d above. In summary, responder 102' can perform an elliptic curve point addition step 424 for the two values or points of L(a) 421a and L(b) 422b. As contemplated herein, the output of an ECDH key exchange function such as 401a or 401c can comprise a point value comprising an X coordinate or value and a Y coordinate or value, and when the output is described herein as a key or a value, the key or value can comprise either (i) the X value only for use in a key derivation function such as 402a, 402b, or 402c, or (ii) the X and Y values together for other calculations such as an ECC point addition step 424. The combination of the points L(a) 421a and L(b) for an ECC point addition step 424 can comprise the point or value L 412, as depicted and described in connection with a key derivation step 423 in Figure 4d. Key derivation step 423 can then input the value L 412, M 410 from message 503, and N 411 from step 314b into key derivation function 402c in order output symmetric ciphering key ke 327a.

Responder 102' can then perform an encryption step 322, where encryption step 322 was depicted and described in connection with Figure 4c above. Step 322 can also include responder 102' creating a value R-auth 404, where R-auth 404 can comprise a hash of the initiator nonce 317a from plaintext 403a in step 320, the second random number R-nonce 318b from a step 318 above, and public PKI keys. The value for R-auth 404 in a step 322 can comprise the value in point 3 of section 6.2.3 of DPP specification version 1.0. An encryption step 322 can comprise responder 102' (a) encrypting the value for R-auth 404 using the mutually derived key ke 327a from the key derivation step 423 above, where (b) the output of encryption step 322 can comprise ciphertext 324a.

Responder 102' can then conduct an encryption step 315b, where the details for an encryption step 315b are depicted and described in connection with Figure 4b above. Ciphertext 324a from an encryption step 322 in the paragraph above can also be encrypted again for an encryption step 315b. The symmetric ciphering key for encryption step 315b can comprise symmetric key k2 314c from a

key exchange step 314b. The output from an encryption step 315b can be ciphertext 315c, as depicted in Figure 4b above.

Responder 102' can conduct a step 324 before sending a message 123. For a step 324, responder 102' can record the derived responder ephemeral public key Pr 102p and responder ephemeral private key pr 102r and initiator bootstrap public key Bi 101a in an initiator table 102u from Figure 1f along with the identity of device 106 comprising potentially MAC.I 106n. In other words, an initiator table 102u can be updated to include information or data (i) received from network 107 (and DPP server 103) regarding devices 106 communication through network 107 via responder 102', and (ii) derived by responder 102' when performing key exchange steps 401a and 401c and key derivation steps 402a, 402b, and 402c. In other words, an initiator table 102u can include columns for the values of M 410, N 411, L 412, L(a) 421a, L(b) 422a, key k1 314a, key k2 314c, and key ke 327a for each device 106 with an identity such as MAC.I 106n. The recording of values for a device 106 or a plurality of devices 106 in an initiator table 102 can be equivalent to the values recorded for a server database 103a depicted and described in connection with Figure 3c above. In this manner, responder 102' can track keys and values for devices 106 since responder 102' may communicate with a plurality of different devices 106 and initiators 101 over time. In exemplary embodiments, responder 102' derives a new responder ephemeral PKI key pair for each unique DPP authentication request message 122 received. Data or values stored by responder 102' pertaining to derived encryption keys (e.g. k1, k2, and ke) and ephemeral PKI keys (e.g. Pr 102p and pr 102r) are preferably erased from an initiator table 102u upon successful completion of a configuration step 106a by device 106.

Responder 102' can then send initiator 101 a DPP authentication response message 123. In exemplary embodiments, message 123 is sent as a unicast message from (i) the responder MAC address MAC.R 102n (which received message 122) to (ii) the initiator MAC address MAC.I 106n (which sent message 122), over (iii) WiFi link 192. As depicted in Figure 5a, message 123 can include hash values of the bootstrap public keys H(Br 102a), H(Bi 101a), the responder ephemeral public key Pr 102p, and the ciphertext 315c. The ciphertext 315c can have at least encrypted values for (i) R-nonce 318b, (ii) I-nonce 317a, (iii) R-capabilities 318c, and (iv) ciphertext 324a. The ciphertext 315c can be encrypted with mutually derived symmetric encryption key k2 314c. Ciphertext 324a can be encrypted with the mutually derived symmetric key ke 327a and can include a plaintext value for R-auth 404. As described above, R-auth 404 can comprise a hash value over at least an initiator nonce 317a received and the responder nonce 318b.

Initiator 101 can receive DPP authentication response message 123 from responder 102' and conduct a series of steps in order to process the message and reply to the responder 102'. Initiator 101 can conduct a step 331, and a step 331 can comprise the series of steps 325, 319a, 320b, 327, 328, and 322b. The series of steps 331 for an initiator 101 to process a DPP authentication response 123 were depicted and described in connection with Figure 3a above. After completing the series of steps for a

step 331 in Figure 5a, initiator 101 can send responder 102' a DPP authentication confirm message 123a. The data and steps to determine and calculate an DPP authentication confirm message 123a for a step 331 can be compatible with section 6.2.4 of DPP specification version 1.0. Note that although the DPP authentication confirm message 123a can be equivalent to the DPP authentication confirm message from section 6.2.4 of DPP specification version 1.0, an important difference can be the operation of responder 102' in Figure 5a versus a responder as contemplated by DPP specification version 1.0. In the present disclosure as depicted in Figure 5a, responder 102' does not record the responder bootstrap private key br 102b.

Responder 102' can listen for either (i) a broadcast message with the hash 199e values for the bootstrap public keys 102aa and 101aa, or (ii) a unicast message to MAC.R 102n from MAC.I 106n for initiator 101 operating in device 106. Responder 102' can receive the DPP authentication confirm message 123a and process the message. DPP authentication confirm message 123a can include an encrypted value for initiator authentication value 404a as a ciphertext 324b, where ciphertext 324b is encrypted with derived key ke 327a in a step 322b within step 331. Upon receipt of DPP authentication confirm message 123a from device 106, responder 102' can first conduct a decryption step 328b, where decryption step 328b can use the same steps as a decryption step 328 as depicted in Figure 4c above, with a difference being decryption step 328b operates on the ciphertext 324b containing the encrypted initiator authentication value I-auth 404a instead of the R-auth 404 value. After a step 328b, responder 102' can read the plaintext initiator authentication value I-auth 404a.

In step 330, responder 102' can compare the decrypted value for I-auth 404a with a calculated value for I-auth 404a and if the two values match then the initiator 101 can be considered by responder 102' to be authenticated. The calculation of I-auth 404a for responder 102' in a step 330 can comprise a calculation of a secure hash value over the responder nonce 318b and bootstrap and ephemeral public keys, with an exemplary calculation for I-auth 404a shown in the sixth paragraph of page 50 of DPP specification version 1.0. Other calculations of an initiator authentication value over the responder nonce 318b are possible as well, without departing from the scope of the present disclosure.

After authentication step 330 is completed, in a step 505, responder 102' can determine the source of network credentials 109 or encrypted network credentials 109' for initiator 101. In a step 505, responder 102' can use the selected responder mode 306a as depicted and described in connection with Figure 3c. For embodiments where a network 107, an external network 108, or DPP server 103 can be the source of network credentials 109 or encrypted network credentials 109' (as specified in a responder mode 306a), then responder 102' can use the series of steps and messages in Figure 6 in order to (i) receive network credentials 109 or encrypted network credentials 109' from a DPP server 103 and (ii) send the credentials to initiator 101. For other embodiments where a responder 102' already records the network credentials 109 for initiator 101 and device 106 (such as responder 102' uses network credentials 109 to connect with network access point 105), then responder 102' can read

the locally stored credentials 109 and send the credentials 109 to initiator 101 in a DPP configuration response message 124 in Figure 6 below. Responder 102' can use a step 349 as depicted in Figure 3b in order to send locally stored credentials 109 to initiator 101, where responder 102' performs the actions instead of a DPP server 103. As depicted in Figure 5a, the collection of steps and messages beginning with responder 102' sending message 123 through step 505 can comprise a step 506. Step 506 can subsequently be used by a responder 102' for a different responder mode 306a depicted in Figure 5b below.

Figure 5b

Figure 5b is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device database, a DPP server, a responder proxy, and an initiator, in accordance with exemplary embodiments. System 500b can include a device database 104, DPP server 103, responder proxy 102', and initiator 101 operating in device 106. Before starting steps and message flows depicted in Figure 5b, the initiator, DPP server, and device database depicted for system 500b in Figure 5b may previously complete exemplary message flows and steps depicted for a step 301 in Figure 3a above. Responder proxy 102' can communicate with DPP server 103 via IP network 113, also as depicted in Figure 3a above.

A difference between system 500b in Figure 5b and system 500a in Figure 5a can be that (i) responder ephemeral PKI keys pr 102r and Pr 102p are derived and recorded by a responder 102' (instead of a DPP server 103 in Figure 3a), (ii) responder 102' sends responder ephemeral private key pr 102r to DPP server 103, and (iii) DPP server 103 derives key L 412 instead of responder 102' in Figure 5a. In exemplary embodiments depicted in Figure 5b the responder bootstrap private key br 102b can remain recorded in a DPP server 103 and not exposed to responder 102'. The exemplary embodiment depicted in Figure 5b can comprise a third mode 306a of operation for responder 102' and DPP server 103, as depicted and described in connection with Figure 3c above.

As depicted in Figure 5b, responder 102' and DPP server 103 can conduct a step 501, where a step 501 is depicted and described in connection with Figure 5a above. Responder 102' can use data received from initiator 101 in a message 122 from step 301 in order to conduct step 501. In summary, a step 501 in Figure 5b can comprise, (i) a step 502 for DPP server 103 to select a mode 306a for both responder 102' and DPP server 103, (ii) DPP server 103 conducting an ECDH key exchange to calculate the value or key M 410, (iii) responder 102' receiving the key M 410 from DPP server 103, (iv) responder 102' deriving the responder ephemeral PKI keys of pr 102r and Pr 102p, (v) responder 102' performing a key derivation step 402a with M 410 in order to calculate key k1 314a, (vi)

responder 102' decrypting ciphertext in message 122 using key k1 314a, and (vii) responder 102' conducting a key exchange step 320 to calculate key k2 314c.

Note that the selected responder mode 306a in a step 501 is not normally observable to an initiator 101, since the full and complete set of DPP 191 messages between initiator 101 and responder 102' can be transmitted and received between the two nodes in Figure 5b. In other words, the different distribution of PKI and shared secret keys in a server database 103a and responder 102' for the responder mode 306a used for Figure 5b would not normally be observable to an initiator 101. The recording of PKI and shared secret keys in a server database is depicted in a server database 103a in Figure 3c, where the value "--" can mean that the key is recorded and/or operated by responder 102'.

Upon completion of a step 501 in Figure 5b, responder 102' can send DPP server 103 a message 507 via secure session 302b established during step 301 from Figure 3a. Message 507 can include (i) an identity for device 106 and initiator 101, such as MAC.I 106n or hash value 101aa, and (ii) the derived responder ephemeral private key pr 102r. DPP server 103 can receive message 507 and use the received device identity to select the initiator bootstrap public key Bi 101a and responder bootstrap private key br 102b from a server database 103a. DPP server can use the received responder ephemeral private key pr 102r from message 507 and the selected data in order to conduct an ECDH key exchange step 401c as depicted and described in connection with Figure 4c above. The output of a step 401c can be key L 412.

After deriving key L 412, DPP server 103 can then send responder 102' a message 508, where message 508 can include an identity for device 106 such as MAC.I 106n (or hash value 101aa), the initiator bootstrap public key Bi 101a (from a server database 103a), and the derived key L 412 from a step 402c. Responder 102' can receive message 508 and record and process the data. Responder 102' can conduct a key derivation function 402c using (i) the received key L 412, (ii) the received key M 310 from a message 503 in a step 501, and (ii) the key N 311 derived by responder 102 in a step 314b in step 501. The output of a key derivation function 402c can be key ke 327a, as depicted and described for a key exchange step 402c in Figure 4c.

Responder 102' can then calculate a responder authentication value 404 and conduct an encryption step 322 using the plaintext 403c as depicted and described for a step 322 in Figure 4c. The output of an encryption step 322 can comprise ciphertext 324a. Responder 102' can then assemble plaintext 403b (which includes ciphertext 324a in the plaintext values for plaintext 403b) and conduct an encryption step 315b, where encryption step 315b is depicted and described in connection with Figure 4b above. Responder 102' can conduct a step 324 before sending a message 123. For a step 324, responder 102' can record the derived responder ephemeral public key Pr 102p and responder ephemeral private key pr 102r and initiator bootstrap public key Bi 101a in an initiator table 102u from Figure 1f along with the identity of device 106 comprising potentially MAC.I 106n.

Responder 102' and initiator 101 can then conduct a step 506, where the series of steps and messages for a step 506 were depicted and described for a responder 102' and initiator 101 in Figure 5a above. In summary for a step 506 depicted above in a Figure 5a, a responder 102' can send initiator 101 a DPP authentication response message 123 via WiFi link 192. Initiator 101 can process the DPP authentication response message 123 using a step 331, which was depicted and described in connection with Figure 3a above. Initiator 101 can then send a DPP authentication confirm message 123a and responder 102' can receive the message 123a via WiFi link 192. Responder 102' can decrypt ciphertext 324b from message 123a using a decryption step 328b in order to read plaintext initiator authentication value 404a. In step 330 of a step 506, responder 102' can compare the decrypted value for I-auth 404a with a calculated value for I-auth 404a and if the two values match then the initiator 101 can be considered by responder 102' to be authenticated.

After authentication step 330 in step 506 is completed, in a step 505 of a step 506, responder 102' can determine the source of network credentials 109 or encrypted network credentials 109' for initiator 101. For embodiments where a network 107, an external network 108, or DPP server 103 can be the source of network credentials 109 or encrypted network credentials 109' (as specified in a responder mode 306a), then responder 102' can use the series of steps and messages in Figure 6 in order to (i) receive network credentials 109 or encrypted network credentials 109' from a DPP server 103 and (ii) send the credentials to initiator 101. For other embodiments where a responder 102' already records the network credentials 109 for initiator 101 and device 106 (such as responder 102' uses network credentials 109 to connect with network access point 105), then responder 102' can read the locally stored credentials 109 and send the credentials 109 to initiator 101 in a DPP configuration response message 124 in Figure 6 below. Responder 102' can use a step 349 as depicted in Figure 3b in order to send locally stored credentials 109 to initiator 101, where responder 102' performs the actions instead of a DPP server 103.

Figure 6

Figure 6 is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a DPP server, a responder proxy, and an initiator, in accordance with exemplary embodiments. System 600 can include a DPP server 103, responder proxy 102', and initiator 101 operating in device 106. Before starting steps and message flows depicted in Figure 6, DPP server 103, responder 102', and initiator 101 can previously complete exemplary message flows and steps depicted for the nodes in either Figure 5a or Figure 5b. System 600 depicts the transfer of credentials 109 or 109' from responder 102' when responder 102' calculates and operates with symmetric ciphering key ke 327a. Although the source of credentials 109 or 109' is depicted as network 107 with DPP server 103 in Figure 6, the source of credentials 109 or 109' could be either

DPP server 103 or responder 102'. A difference between system 600 and system 300b in Figure 3b is that for system 300b, DPP server 103 calculates, records, and operates with key ke 327a for use with initiator 101 in device 106.

After transmitting or sending DPP authentication confirm message 123a in a step 506 as depicted and described in connection with Figure 5a and Figure 5b, initiator 101 can conduct a step 332. Initiator 101 can take on a roll of enrollee, such that initiator 101 can prepare to send a configuration attribute 330a. Configuration attribute 330a can be used to specify that initiator 101 can take the roll of enrollee and is expecting a set of network credentials 109 from responder 102', with associated data for responder 102' to prepare or format a set of network credentials 109.

In step 332, initiator 101 can also generate a random number, for an enrollee nonce E-nonce 332a. E-nonce 332a and other nonce values can be processed using nonce parameters 199g. Initiator 101 can then conduct an encryption step 322c, where encryption step 322c can comprise (i) an encryption step 322 using key ke 327a, and (ii) the plaintext to be encrypted can be the E-nonce 332a and the configuration attribute 330a. The output of an encryption step 322c can be a ciphertext 324c, which is also described for a step 322c in Figure 3b above. Initiator 101 can then send via WiFi link 192 to responder 102' a DPP Configuration Request message 123b, where the configuration request 123b can include the ciphertext 324c. The plaintext in ciphertext 324c can include E-nonce 332a and the configuration object 330. DPP Configuration Request message 123b can be processed by initiator 101 according to section 6.3.3 of DPP specification version 1.0.

Responder 102' can receive DPP Configuration Request message 123b via connection 192 and using DPP application 102g. Responder 102' can conduct a decryption step 328c using decryption step 328 depicted in Figure 4c above. A difference between decryption step 328 and decryption step 328c is that decryption step 328c uses the ciphertext 324c as input into the symmetric decryption algorithm 405b. Responder 102' can use the previously calculated key ke 327a in order to read the plaintext within ciphertext 324c. Responder 102' can record the plaintext E-nonce 332a and configuration attribute 330a.

Responder 102' can then conduct a step 601 in order to determine the source of credentials 109 for initiator 101 and device 106. For embodiments where responder 102' previously records a valid set of network credentials 109 for use with network access point 105 depicted in Figure 1a, then responder 102' could select the recorded network access credentials 109 for use as a configuration object for initiator 101 and device 106. In other words, for embodiments where responder 102' previously connects with network access point 105 (or responder 102' could operate within network access point 105), then responder 102' could record a valid set of network credentials 109 in order to successfully connect with network access point 105. For these embodiments, responder 102' could select the recorded network access credentials 109 for transfer to initiator 101 and used by initiator 101 and device 106 in order to also connect with access point 105 using the same set of network

credentials 109 recorded by responder 102'. Responder 102' can read the locally stored credentials 109 and send the credentials 109 to initiator 101 in a DPP configuration response message 124 in Figure 6. Responder 102' can use a step 349 as depicted in Figure 3b in order to send locally stored credentials 109 to initiator 101, where responder 102' performs the actions instead of a DPP server 103.

For other embodiments, such as responder 102' does not previously record a set of network credentials 109 for use by initiator 101 and device 106, then in a step 601 responder 102' could determine a query to DPP server 103 or network 107 is required for the network credentials 109 or encrypted network credentials 109' for initiator 101 and device 106. The determination of a source for network credentials (e.g. either responder 102' or DPP server 103 could also be specified in a responder mode 306a, such that a first responder mode 306a selects responder 102' as the source of credentials 109 and a second responder mode 306a selects DPP server 103 as the source of credentials 109 or encrypted credentials 109'). Other possibilities exist as well for a step 601 by responder 102' in Figure 6 to determine a source of credentials 109 for device 106 without departing from the scope of the present disclosure.

For embodiments where step 601 by responder 102' determines that network 107 or DPP server 103 is the source of network credentials 109 for device 106, responder 102' can then send DPP server 103 a message 602 through secure session 302b (depicted and described in Figure 3a), where message 602 include an identity for device 106 such as MAC.I 106n (or device identity 106i or hash value 101aa) and the plaintext configuration attribute 330a. DPP server 103 can receive message 602 and can conduct a step 337 in order to obtain network credentials 109 or asymmetrically encrypted credentials 109'. A step 337 for DPP server 103 was depicted and described in connection with Figure 3b above, and the same step can be used by DPP server 103 for system 600 in Figure 6 (for embodiments where step 601 by responder 102' determines that network 107 or DPP server 103 is the source of network credentials 109 for device 106).

Continuing with embodiments where step 601 by responder 102' determines that network 107 or DPP server 103 is the source of network credentials 109 for device 106, DPP server 103 can then send responder 102' a message 603, where message 603 includes (i) an identity for device 106 such as MAC.I 106n or ID.device 106i or hash value 101aa, and (ii) a set of network credentials 109 or asymmetrically encrypted network credentials 109'. Asymmetrically encrypted credentials 109' could be in message 603 if DPP server 103 obtained the credentials outside of or external to DPP server 103, such as an external network 108 (where external network 108 may prefer that only device 106 be able to read the plaintext network credentials 109 inside asymmetrically encrypted network credentials 109'). A difference between message 603 in Figure 6 and message 341 in Figure 3b can be (i) the message 603 is not encrypted with key ke 327a, and (ii) the message 603 may not include a digital signature 338c. Or, for other embodiments, then a system 600 could include the use of a digital

signature creation step 338 as depicted and described in connection with Figure 3b and Figure 3d and the message 603 and subsequent message 124 can include the digital signature 338c. Responder 102' can receive message 603 and process the data. Note that message 603, step 337, and message 603 can be omitted for embodiments where step 601 determines that responder 102' is the source of credentials 109 for device 106.

For embodiments where message 603 does not include a digital signature 338c as depicted in Figure 6, responder 102' can then record the network credentials 109 or asymmetrically encrypted network credentials 109' and conduct a signature creation step 338. Note that the source of credentials 109 can be either responder 102' or DPP server 103, depending on the source selected in step 601. In exemplary embodiments, asymmetrically encrypted network credentials 109' can be from DPP server 103 or potentially an external network 108. For a step 338 in Figure 6, responder 102' can conduct a signature creation step 338 as depicted and described in connection with Figure 3d above. The configuration object with network credentials 109 (or encrypted credentials 109') can be signed using the recorded responder ephemeral private key pr 102r, corresponding to public key Pr 102p sent in message 325 in Figure 3a above. The digital signature by responder 102' in Figure 3b can comprise a digital signature 338c. Or, in another embodiment the configuration object with network credentials 109 can be signed using the recorded responder bootstrap private key br 102b recorded by DPP server 103 and digital signature 338c included in a message 603, where initiator 101 could verify the signature 338c with the bootstrap public key Br 102a.

At step 332d, responder 102' can conduct an encryption step 332d using an encryption step 322 depicted and described in connection with Figure 4c above. Encryption step 332d can be performed using key ke 327a for device 106 and initiator 101 from an initiator table 102u. For encryption step 332d, the plaintext values of (i) the E-nonce 332a and (ii) the configuration object comprising network credentials 109 (or asymmetrically encrypted credentials 109') can be input into a symmetric encryption algorithm 405a in order to generate a ciphertext 324d. As depicted, the collection of steps and messages from 602 through 322d can comprise a step 604. In some exemplary embodiments using a fourth responder mode as described in Figure 3c where responder 102 operates as a responder 102* using conventional technology in Figure 1b and Figure 1c, then a responder 102* could use the step 604 in order to obtain the credentials 109 or 109' from a DPP server 103.

Responder 102' can then send initiator 101 a DPP configuration response message 124 via connection 192, where message 124 can include the ciphertext 324d, and the ciphertext 324d can include the E-nonce 332a, the signature 338c, and either network credentials 109 or asymmetrically encrypted network credentials 109'. If network credentials 109' are sent in ciphertext 324d, then the network credentials 109' can comprise a set of "double encrypted" network credentials of (i) asymmetrically encrypted network credentials 109' and (ii) then encrypted again in a ciphertext 324d using the key ke 327a.

Initiator 101 can receive message 124 from responder 102' with ciphertext 324d. Initiator 101 can conduct a decryption step 328d using decryption step 328 depicted in Figure 4c above. A difference between decryption step 328 and decryption step 328d is that decryption step 328d uses the ciphertext 324d as input into the symmetric decryption algorithm 405b. Initiator 101 can use the previously calculated key ke 327a from a step 331 in order to read the plaintext within ciphertext 324d. Initiator 101 can confirm that plaintext in ciphertext 324d includes the previously sent E-nonce 332a. Initiator 101 can extract a plaintext value for (i) network credentials 109 or (ii) asymmetrically encrypted network credentials 109' using symmetric decryption algorithm 405b and key ke 327a. In other words, with credentials 109' decryption step 328d removes the first layer of "double encryption" where the first layer was encrypted with key ke 327a, and the "plaintext" after decryption step 328d for credentials 109' can comprise the asymmetrically encrypted network credentials 109' (e.g. with the first layer of encryption with key ke 327a has been removed).

Initiator 101 can conduct a signature verification step 340 as depicted and described in connection with Figure 3d above in order to verify signature 338c received in message 124. Initiator 101 can use either (i) responder ephemeral bootstrap public key Pr 102p or (ii) responder bootstrap public key Br 102a in order to conduct a signature verification step 340 on digital signature 338c. The use of either (i) responder ephemeral bootstrap public key Pr 102p or (ii) responder bootstrap public key Br 102a can be specified in a set of parameters 199 or as part of asymmetric encryption parameters 199h or digital signature parameters 199i. Note that if responder bootstrap public key Br 102a is used to verify digital signature 338c in a step 340, then DPP server 103 would create digital signature 338c in a step 338 with responder bootstrap private key br 102b. If plaintext network credentials 109 are included in ciphertext 324f, then initiator 101 can read the plaintext values for credentials 109. If asymmetrically encrypted network credentials 109' are included in ciphertext 324d, then initiator 101 or device 106 operating initiator 101 can conduct an asymmetric decryption step 339 depicted above in Figure 3e in order to convert credentials 109' into plaintext credentials 109.

After reading the plaintext network credentials 109 and optionally verifying a signature using a recorded public key 102p or 102a, device 106 can complete the configuration step 106a by loading the network credentials 109 and begin operating a WiFi radio 106r with the network credentials 109. In other words, the WiFi data link 192 depicted in Figure 1a can be converted into a fully authenticated and encrypted WiFi connection using network credentials 109 and standards such as WPA2, WPA3, and also subsequent or related standards where WiFi nodes such as clients or access points use a compatible set of network credentials 109. For example, device 106 could begin scanning for SSID.network-AP 109a on a channel in Config.network-AP 109c (which can be included in credentials 109 or 109'), and then send access point 105 a probe request. In this manner, device 106 can obtain connectivity to network access point 105 through a configuration step 106a.

Figure 7

Figure 7 is a graphical illustration of an exemplary system, where an access point with an initiator communicates with a responder proxy using a hosted device provisioning protocol, in accordance with exemplary embodiments. The present disclosure contemplates that the systems and methods for using a responder 102' with a DPP server 103 can support a device provisioning protocol 191 in order to configure a network access point 105'. System 700 in Figure 7 can include an unconfigured network access point 105', a mobile phone 701, and a network 107. Unconfigured network access point 105' can include an initiator 101, where initiator 101 can have the same or equivalent functionality as an initiator 101 depicted and described above in Figure 1a, Figure 1b, Figure 1c, Figure 3a, Figure 3b, Figure 5a, Figure 5b, and Figure 6.

Initiator 101 can include a responder bootstrap public key Br 102a, an initiator bootstrap private key bi 101b, and an initiator bootstrap public key Bi 101a. PKI keys could be recorded in access point 105' before access point 105' is installed at an operating location, such as during manufacturing or distribution of unconfigured access point 105'. Initiator 101 operating in unconfigured access point 105' can have responder capabilities of an enrollee. Before installation of access point 105' at an operating location, initiator 101 could record a responder public key table 101ta (depicted and described in connection with Figure 1d and also Figure 1e) with a plurality of responder bootstrap public keys Br 102a for a plurality of different networks 107.

In exemplary embodiments, the responder public key table 101ta could be loaded into access point 105' during manufacturing or distribution of access point 105' or a configuration step for access point 105' before conducting DPP 191. As contemplated herein, an access point 105 or access point 105' can comprise a "WiFi router", where one interface is a WiFi radio similar to 106r or 102r and the other interface is a wired connection such as Ethernet, DSL, or a coaxial cable. For some exemplary embodiments where a radio 102x uses wireless technology other than WiFi, then access point 105' could comprise a "gnodeb" for a 5G network, and "enodeb" for a 4G LTE network, and other possibilities exist as well for the wireless technology used by a radio 102x in access point 105'.

Unconfigured access point 105' can power up and conduct a step 200 from Figure 2a and Figure 3a in order to periodically transmit DPP authentication request messages 122 using different recorded responder bootstrap public keys (combined with possibly different initiator bootstrap PKI keys) from an initiator key database 101t in order to "search" or "scan" for a compatible responder to communicate with. When mobile phone 701 is taken to the vicinity or within WiFi range of AP 105' and activates DPP application 102g and responder 102', then responder 102' can receive the DPP authentication request message 122 and begin processing the message if mobile phone 701 records a matching hash value 102aa over at least the hash over responder bootstrap public key Br 102a recorded in memory, such as in a responder network table 102t depicted in Figure 1f.

Mobile phone 701 can comprise a smart phone or a mobile handset, such as a mobile phone manufactured by Apple® or running an Android based operating system. Mobile phone 701 could also comprise another mobile and networked computing device, such as a laptop or a tablet. Mobile phone 701 could include a responder 102', where responder 102' could have the same or equivalent functionality as responder 102' depicted and described in connection with Figure 1a, Figure 1d, Figure 1f, Figure 3a, etc. In exemplary embodiments, responder 102' can operate as an application for mobile phone 701 downloaded from a repository of mobile phone applications, such as an "app store" or "play store", or possibly a private application. In other words, embodiments herein contemplate that a mobile phone 701 could comprise the physical entity operating or processing a responder 102' and the software or firmware instructions for operation of responder 102' could comprise a DPP application 102g. Other physical arrangements are possible as well, without departing from the scope of the present disclosure. Mobile phone 701 can have both a WiFi radio 102x and a WAN radio 102h. Mobile phone 701 can communicate with unconfigured access point 105' via the WiFi radio 102x and a network 107 and a DPP server 103 via the WAN radio 102h and an IP network 113. Responder 102' for mobile phone 701 can have capabilities of a configurator.

Network 107 can operate a DPP server 103, where a DPP server 103 in Figure 7 can have the functionality of a DPP server 103 depicted and described in connection with Figures 1a, Figure 3a, Figure 3b, Figure 3c, Figure 5a, Figure 5b, and Figure 6. Although not depicted in Figure 7, a network 107 can also operate other supporting servers and/or running processes depicted in Figure 1a, Figure 2a, etc., such as a device database 104. Network 107 could be connected to mobile phone 107 via the IP network 113 depicted in Figure 1a and using a secure session 302b from Figure 3a.

As depicted in Figure 7, mobile phone 107 using responder 102' could conduct the series of messages for a device provisioning protocol 191 with initiator 101 operating in unconfigured access point 105'. The series of messages for a device provisioning protocol 191 could comprise the messages 122, 123, 123a, 123b, and 124 as depicted and described in connection with Figures 1a, 3a, 3b, 5a, 5b, and 6 above. In order for responder 102' to send and receive the device provisioning protocol messages 191 depicted in Figure 7, responder 102' transmits and receives a series of messages with DPP server 103 in network 107, where the series of messages could comprise a hosted device provisioning protocol 702. For a first mode 306a, representing embodiments depicted in Figure 3a (where responder bootstrap and ephemeral secret keys remain in DPP server 103), the series of messages for a hosted device provisioning protocol 702 could comprise the messages between responder 102' and DPP server 103 that are depicted in Figure 3a and Figure 3b (e.g. messages 302b, 305, 309, 312b, etc.).

For a second mode 306a, representing embodiments depicted in Figure 5a where the responder bootstrap private key br 102b can remain in DPP server 103 (but DPP server 103 does not operate on responder ephemeral private key pr 102r), the series of messages for a hosted device provisioning

protocol 702 could comprise the messages between responder 102' and DPP server 103 that are depicted in Figure 5a (e.g. messages in step 301 and then 503, 504, etc.). For a third mode 306a, representing embodiments depicted in Figure 5b where (i) the responder bootstrap private key br 102b remains in DPP server 103 and (ii) responder 102' derives key pr 102r and sends the key to DPP server 103, the series of messages for a hosted device provisioning protocol 702 could comprise the messages between responder 102' and DPP server 103 that are depicted in Figure 5b (e.g. messages in step 301 and then message 507, 508, etc.).

Upon completion of a device provisioning protocol 191 and a hosted device provisioning protocol 702, unconfigured network access point 105' can conduct a configuration step 106a to load a set of network credentials 109 and begin operating with the credentials 109, as depicted in Figure 7. The configuration attribute in a message 123b can specify that access point 105' is to be configured as an access point (or with a value of "ap"). The configuration step 106a can convert the unconfigured network access point 105' into an operating network access point 105, such that WiFi clients (or "sta") can connect with network access point 105 using a compatible set of network credentials 109. As depicted in Figure 7, (i) a mobile phone 701 could also operate as a responder 102' with a roll of a configurator and (ii) the access point 105' operating with a roll of enrollee.

As depicted in Figure 7, mobile phone 701 could also conduct a device provisioning protocol 191 using a hosted device provisioning protocol 702 with DPP server 103 in order to transfer a set of compatible network credentials 109 to device 106, such that device 106 could connect with network access point 105. The compatible network credentials 109 for device 106 can specify that device 106 operates in a WiFi client mode with network access point 105. Other possibilities exist as well for the operation of a responder 102' to support the transfer of credentials to devices operating as initiators 106 without departing from the scope of the present disclosure.

Figure 8

Figure 8 is a graphical illustration of an exemplary system, where an access point operates as a responder proxy for a device with an initiator, in order transfer a set of network credentials to the device, in accordance with exemplary embodiments. The present disclosure contemplates embodiments where a network access point 105 can operate as a responder 102' for a device 106 operating as an initiator 101, in order for the network access point 105 to transfer network credentials 109 to the device 106. System 800 in Figure 8 can include an unconfigured device 106', a network access point 105, and a network 107.

Network access point 105 in a system 800 can operate with a set of network access credentials 109 and also operate as a responder 102'. Note that network access point 105 as a WiFi router can simultaneously support other devices routing WiFi data through network access point 105

concurrently with network access point 105 operating as a WiFi router. In other words, the present disclosure contemplates that when a radio such as WiFi radio 102x operates for a responder 102' listening for incoming DPP authentication request messages 122, the radio 102x can concurrently support regular WiFi communication according to standard protocols such as WPA2, WPA3, etc.

5 Network access point 105 can connect with a network 107 via an IP network 113 and a secure session 302b as depicted and described in connection with Figure 3a. Network 107 can operate a DPP server 103, where the function and operation of a DPP server 103 is depicted and described above in connection with Figure 1a, Figure 2a, Figure 3a, etc. Although not depicted in Figure 8, network 107 could also operate a device database 104 and DPP server 103 could include a server database 103b.

10 Network access point 105 could comprise a WiFi router and operate a WiFi radio and communicate with a plurality of devices 106 that have already performed a configuration step 106a or a manual configuration to connect the configured devices 106. For some exemplary embodiments where a radio 102x uses wireless technology other than WiFi, then access point 105' could comprise a "gnodeb" for a 5G network, and "enodeb" for a 4G LTE network, and other possibilities exist as well for the wireless

15 technology used by a radio 102x in access point 105'.

Network access point 105 can have the internal electrical components equivalent to a responder 102' depicted and described above in connection with Figure 1f, including a CPU, RAM, nonvolatile memory, and a system bus. For the embodiment depicted in Figure 8, the responder 102' in network access point 105 could comprise software, firmware, or a running process within network

20 access point 105. Responder 102' in network access point 105 could have the same or equivalent functionality as responder 102' depicted and described in connection with Figure 1a, Figure 1f, Figure 3a, Figure 3b, Figure 5a, Figure 5b, and Figure 6. Network access point 105 could operate with capabilities as a configurator 801, as depicted in Figure 8.

An unconfigured device 106' could be a device with a WiFi radio 106r and operate an initiator

25 101. The electrical components inside device 106' could be similar to the electrical components depicted for a responder 102' in Figure 1f, such as including a CPU, RAM, nonvolatile memory, an operating system, and a system bus. Initiator 101 in unconfigured device 106' could have the same or equivalent functionality as a initiator 101 depicted and described above in Figure 1a, Figure 1b, Figure 1c, Figure 1d, Figure 1e, Figure 3a, etc.. Unconfigured device 106' can have capabilities as an enrollee

30 802.

Unconfigured device 106' can power up and conduct a step 200 from Figure 2b and Figure 3a in order to periodically transmit DPP authentication request messages 122 using different recorded responder bootstrap public keys (combined with possibly different initiator bootstrap PKI keys) from an initiator key database 101t in Figure 1d in order to "search" or "scan" for a compatible responder

35 to communicate with. When unconfigured device 106' is powered on or taken to the vicinity or within WiFi range of AP 105, then responder 102' in WiFi access point 105 can receive the DPP

authentication request message 122 from unconfigured device 106' and begin processing the message 122 if WiFi access point 105 records a matching hash value 102aa over the hash over responder bootstrap public key Br 102a recorded in memory, such as in a responder network table 102t depicted and described in connection with Figure 1f.

5 As depicted in Figure 8, network access point 105 with responder 102' could conduct the series of messages for a device provisioning protocol 191 with initiator 101 operating in unconfigured device 106'. The series of messages for a device provisioning protocol 191 could comprise the messages 122, 123, 123a, 123b, and 124 as depicted and described in connection with Figure 3a, Figure 3b, Figure 5a, Figure 5b, and Figure 6. In order for responder 102' in WiFi access point 105 to send and
10 receive the device provisioning protocol messages 191 depicted in Figure 8, responder 102' (or AP 105 operating the responder 102') transmits and receives a series of messages with DPP server 103 in network 107, where the series of messages could comprise a hosted device provisioning protocol 702. The series of messages for a hosted device provisioning protocol 702 were depicted and described in connection with Figure 7 above. Note that the steps and messages for conducting a hosted device
15 provisioning protocol 702 can depend on a responder mode 306a, where the responder mode 306a can be selected by a DPP server 103 in a step 306. The present disclosure also contemplates that a responder mode 306a for a hosted device provisioning protocol 702 can be selected by a responder 102'. For these embodiments, where mode 306a is selected by a responder 102', a selected responder mode 306a by the responder 102' could be transmitted from responder 102' to DPP server 103 in a
20 message 305 depicted and described in connection with Figure 3a above.

The combined (i) device provision protocol 191 between network access point 105 and unconfigured device 106' and (ii) hosted device provisioning protocol 702 between network access point 105 and DPP server 103 can securely transfer a set of compatible network access credentials 109 to device 106'. Upon completion of a device provisioning protocol 191 and a hosted device
25 provisioning protocol 702, unconfigured device 106' can conduct a configuration step 106a to (i) load a compatible set of network credentials 109 for network access point 105 and (ii) begin operating with the credentials 109, as depicted in Figure 8. The configuration step 106a can convert the unconfigured device 106' into a configured and operating device 106, such that device 106 can conduct a WiFi connection setup step 803 using the received set of compatible network credentials 109. The WiFi
30 connection setup step 803 could use the series of steps and messages as specified by standards such as WPA2 and WPA3 with credentials 109, and other possibilities exist as well for a WiFi connection setup step 803 without departing from the scope of the present disclosure.

Conclusion

Various exemplary embodiments have been described above. Those skilled in the art will
35 understand, however, that changes and modifications may be made to those examples without departing from the scope of the claims.

CLAIMS

1. A method for conducting a device provisioning protocol (DPP), the method performed by a responder, the method comprising:

a) establishing a secure session with a DPP server using at least a certificate for the DPP server;

b) receiving, from an initiator, a DPP authentication request, wherein the DPP authentication request includes at least an initiator ephemeral public key, a hash value for a responder bootstrap public key, and a first ciphertext;

c) sending, to the DPP server, at least the initiator ephemeral public key, the hash value, and the first ciphertext;

d) receiving, from the DPP server, a responder ephemeral public key and a second ciphertext, wherein the second ciphertext is encrypted using at least a responder ephemeral private key corresponding to the responder ephemeral public key, and wherein the second ciphertext includes at least a responder nonce and a responder authentication value;

e) sending, to the initiator, a DPP authentication response, wherein the DPP authentication response includes at least the received responder ephemeral public key and the second ciphertext; and

f) receiving, from the DPP server, a third ciphertext, wherein the third ciphertext includes network credentials for the initiator, and wherein the third ciphertext is encrypted using at least (i) the responder ephemeral private key, and (ii) the responder bootstrap private key.

2. The method of claim 1, wherein the DPP server derives the responder ephemeral private key and the responder ephemeral public key using a set of cryptographic parameters, and wherein the set of cryptographic parameters are selected using at least the hash value.

3. The method of claim 2, wherein the hash value, the set of cryptographic parameters, and the responder bootstrap private key are stored in a device database, and wherein the DPP server uses the hash value to query the device database for the set of cryptographic parameters and the responder bootstrap private key.

5 4. The method of any one of claims 1 to 3, wherein the responder sends the hash value for the responder bootstrap public key to the DPP server before step b), and wherein the responder stores the hash value before a step a).

5. The method of any one of claims 1 to 4, further comprising, after step a) and before step d), receiving a responder mode from the DPP server, wherein the responder mode signals the
10 DPP server to derive the responder ephemeral private key.

6. The method of any one of claims 1 to 5, wherein the second ciphertext includes the responder authentication value as a fourth ciphertext, and wherein the fourth ciphertext is encrypted by the DPP server using at least (i) the responder ephemeral private key, and (ii) the responder bootstrap private key.

15 7. The method of claim 6, wherein the fourth ciphertext is encrypted using at least an initiator bootstrap public key, and wherein a network operating the DPP server stores the initiator bootstrap public key.

8. The method of any one of claims 1 to 7, wherein the first ciphertext is encrypted with a symmetric ciphering key, and wherein the symmetric ciphering key is derived by the
20 initiator using the responder bootstrap public key and an initiator ephemeral private key corresponding to the received initiator ephemeral public key.

9. The method of any one of claims 1 to 8, wherein the DPP server conducts an elliptic curve Diffie-Hellman key exchange with the responder ephemeral private key and the initiator ephemeral public key in order to derive a symmetric ciphering key for the second ciphertext.

10. The method of any one of claims 1 to 9, wherein the DPP server conducts an elliptic curve Diffie-Hellman key exchange with the responder ephemeral private key, the responder bootstrap private key, and an initiator bootstrap public key in order to derive a symmetric ciphering key for the third ciphertext.

5 11. The method of claim 10, wherein the DPP authentication request includes a second hash value for the initiator bootstrap public key, wherein the responder sends to the DPP server the second hash value in a step c), and wherein the DPP server selects the initiator bootstrap public key for the initiator using the second hash value.

10 12. The method of any one of claims 1 to 11, wherein the responder comprises a responder proxy, and wherein the responder uses a WiFi radio with a responder configuration in order to receive the DPP authentication request message from the initiator.

13. The method of any one of claims 1 to 12, further comprising after step e) and before step f), receiving, from the initiator, a DPP authentication confirm message with an encrypted initiator authentication value, wherein the responder sends to the DPP server at least the encrypted
15 initiator authentication value.

14. The method of claim 13, wherein the DPP server decrypts the encrypted initiator authentication value using at least (i) the responder ephemeral private key, and (ii) the responder bootstrap private key, and wherein the DPP server compares the received initiator authentication value with a calculated initiator authentication value.

20 15. The method of any one of claims 1 to 14, further comprising:
recording a responder network table, wherein the responder network table includes at least the hash value; and
using the hash value and the responder network table to select the DPP server.

16. A method for conducting a device provisioning protocol (DPP), the method performed by a responder, the method comprising:

a) recording a set of cryptographic parameters and a responder network table;

b) receiving a DPP authentication request from an initiator, wherein the DPP authentication request includes at least an initiator ephemeral public key and a hash value for a responder bootstrap public key;

c) using the hash value and the responder network table to select a DPP server;

d) sending at least the initiator ephemeral public key and the hash value to the selected DPP server;

e) deriving (i) a responder ephemeral private key, (ii) a corresponding responder ephemeral public key, and (iii) a responder nonce using the set of cryptographic parameters;

f) receiving an initiator bootstrap public key and a secret key from the DPP server;

g) conducting an elliptic curve Diffie-Hellman (ECDH) key exchange using the derived responder ephemeral private key and the received initiator bootstrap public key in order to derive a first shared secret key;

h) conducting an elliptic curve point addition with the secret key and the derived first shared secret key in order to derive a second shared secret key; and

i) encrypting at least the responder nonce using the derived second shared secret key.

17. The method of claim 16, wherein the responder comprises a responder proxy, and wherein the responder establishes a secure session with the DPP server using at least a certificate for the responder.

18. The method of claim 16 or 17, further comprising (i) transmitting the encrypted responder nonce to the initiator in a DPP authentication response message, and (ii) receiving from the initiator a DPP authentication confirm message.

19. The method of claim 18, further comprising (i) receiving from the initiator a DPP configuration request after receiving the DPP authentication confirm message, and (ii) transmitting a DPP configuration response message, wherein the DPP configuration response message includes network credentials for the initiator.

5 20. The method of claim 19, wherein the responder stores the network credentials.

21. The method of claim 19, further comprising (i) sending at least a configuration attribute from the DPP configuration request to the DPP server, and (ii) receiving the network credentials from the DPP server before sending the DPP configuration response message.

22. The method of claim 21, wherein the network credentials received from the DPP
10 server comprise asymmetrically encrypted network credentials.

23. The method of any one of claims 16 to 22, wherein the responder conducts the ECDH key exchange and the elliptic curve point addition using the set of cryptographic parameters.

24. The method of any one of claims 16 to 23, wherein the responder network table
15 includes at least (i) a plurality of hash values for a plurality of responder bootstrap public keys and (ii) a list of DPP servers.

25. The method of any one of claims 16 to 24, wherein the derived second shared secret key comprises a key "L", wherein the responder derives a symmetric ciphering key using a key derivation function and at least the key "L", and wherein the responder encrypts at least the
20 responder nonce using the symmetric ciphering key.

26. The method of any one of claims 16 to 25, wherein the DPP authentication request includes a first ciphertext of at least an initiator nonce, and wherein the first ciphertext is encrypted with a symmetric ciphering key, and wherein the symmetric ciphering key is derived by the initiator using the responder bootstrap public key and an initiator ephemeral private key
25 corresponding to the initiator ephemeral public key.

27. A method for conducting a device provisioning protocol (DPP), the method performed by a responder, the method comprising:

a) recording a set of cryptographic parameters and a responder network table, wherein the responder network table includes at least (i) a plurality of hash values for a plurality of responder bootstrap public keys and (ii) a list of DPP servers;

b) receiving, from an initiator, a DPP authentication request, wherein the DPP authentication request includes at least an initiator ephemeral public key, a hash value for a responder bootstrap public key, and a first ciphertext;

c) using the hash value and the responder network table to select a DPP server;

d) sending the initiator ephemeral public key, the hash value for a responder bootstrap public key, and the first ciphertext to the selected DPP server;

e) receiving, from the DPP server, a responder ephemeral public key and a second ciphertext, wherein the second ciphertext is encrypted using at least a responder ephemeral private key corresponding to the responder ephemeral public key, and wherein the second ciphertext includes at least a responder nonce and a responder authentication value;

f) sending, to the initiator, a DPP authentication response, wherein the DPP authentication response includes at least the received responder ephemeral public key and the second ciphertext; and

g) receiving, from the DPP server, a third ciphertext, wherein the third ciphertext includes network credentials for the initiator, and wherein the third ciphertext is encrypted using at least (i) the responder ephemeral private key, and (ii) the responder bootstrap private key.

28. The method of claim 27, wherein the responder comprises a responder proxy, and wherein the responder establishes a secure session with the DPP server using at least a certificate for the DPP server.

29. The method of claim 27 or 28, wherein the DPP server derives the responder ephemeral private key and the responder ephemeral public key using the set of cryptographic parameters, and wherein the set of cryptographic parameters are selected by the DPP server using at least the hash value.

5 30. The method of claim 29, wherein the hash value, the set of cryptographic parameters, and the responder bootstrap private key are stored in a device database, and wherein the DPP server uses the hash value to query the device database for the set of cryptographic parameters and the responder bootstrap private key.

31. The method of any one of claims 27 to 30, wherein the responder sends the hash
10 value for the responder bootstrap public key to the DPP server before step b), and wherein the responder network table includes the hash value.

32. The method of any one of claims 27 to 31, further comprising, after step a) and before step e), receiving a responder mode from the DPP server, wherein the responder mode signals the DPP server to derive the responder ephemeral private key.

15 33. The method of any one of claims 27 to 32, wherein the first ciphertext is encrypted with a symmetric ciphering key, and wherein the symmetric ciphering key is derived by the initiator using the responder bootstrap public key and an initiator ephemeral private key corresponding to the initiator ephemeral public key.

34. The method of any one of claims 27 to 33, wherein the DPP server derives the
20 responder ephemeral private key, and wherein the DPP server conducts an elliptic curve Diffie-Hellman key exchange with the responder ephemeral private key and the initiator ephemeral public key in order to derive a symmetric ciphering key for the second ciphertext.

35. The method of any one of claims 27 to 34, wherein the second ciphertext includes the responder authentication value as a fourth ciphertext, and wherein the fourth ciphertext is
25 encrypted by the DPP server using at least (i) the responder ephemeral private key, and (ii) the responder bootstrap private key.

36. The method of claim 35, wherein the fourth ciphertext is encrypted using at least an initiator bootstrap public key, and wherein a network operating the DPP server stores the initiator bootstrap public key.

37. The method of any one of claims 27 to 36, wherein the DPP server conducts an elliptic curve Diffie-Hellman key exchange with the responder ephemeral private key, the responder bootstrap private key, and an initiator bootstrap public key in order to derive a symmetric ciphering key for the third ciphertext.

38. The method of claim 37, wherein the DPP authentication request includes a second hash value for an initiator bootstrap public key, wherein the responder sends to the DPP server the second hash value in a step d), and wherein the DPP server selects the initiator bootstrap public key for the initiator using the second hash value.

39. The method of any one of claims 27 to 38, wherein the responder comprises a responder proxy, and wherein the responder uses a WiFi radio with a responder configuration in order to receive the DPP authentication request message from the initiator.

40. The method of any one of claims 27 to 39, further comprising after step f) and before step g), receiving, from the initiator, a DPP authentication confirm message with an encrypted initiator authentication value, wherein the responder sends to the DPP server at least the encrypted initiator authentication value.

41. The method of claim 40, wherein the DPP server decrypts the encrypted initiator authentication value using at least (i) the responder ephemeral private key, and (ii) the responder bootstrap private key, and wherein the DPP server compares the received initiator authentication value with a calculated initiator authentication value.

42. A system for supporting a device provisioning protocol (DPP), the system comprising:

a wireless device for operating a DPP application, wherein the DPP application comprises at least, in part, a responder, for establishing a secure session with a DPP server using at least a certificate for the DPP server, for storing at least one hash value for a responder bootstrap public key, for using a responder configuration to communicate over a WiFi link with an initiator, for using the WiFi link to receive a DPP authentication request message with at least (i) an initiator ephemeral public key and (ii) the hash value, for receiving an encrypted set of network credentials from the DPP server for the initiator, and for sending over the WiFi link the encrypted set of network credentials to the initiator in a DPP configuration response message;

a database for recording (i) a responder bootstrap private key and (ii) a set of cryptographic parameters for the responder bootstrap private key;

a DPP server for receiving the initiator ephemeral public key and the hash value from the wireless device, for querying the database using the hash value in order to receive the responder bootstrap private key, and for conducting an elliptic curve Diffie-Hellman (ECDH) key exchange using at least (i) the received initiator ephemeral public key, (ii) the received responder bootstrap private key, and (iii) the set of cryptographic parameters in order to derive a shared secret;

a processor in the DPP server for deriving a symmetric ciphering key using at least the derived shared secret, and for encrypting the set of network credentials using the symmetric ciphering key and the set of cryptographic parameters; and

a LAN interface in the DPP server for using the secure connection (i) to send the hash value to the wireless device before the wireless device receives the DPP authentication request and (ii) to send the encrypted set of network credentials to the initiator.

43. The system of claim 42, further comprising the DPP server for selecting a responder mode for the wireless device, and for sending the responder mode to the wireless device.

44. The system of claim 42 or 43, wherein the wireless device operates as a responder proxy, and wherein the wireless device does not receive and does not operate with the responder bootstrap private key.

45. The system of any one of claims 42 to 44, further comprising a network access
5 point for authenticating with the set of network access credentials, and for providing connectivity with an IP network to the initiator after authenticating with the set of network access credentials.

46. The system of any one of claims 42 to 45, further comprising the DPP server for
deriving (i) a responder ephemeral public key for a DPP authentication response message after
receiving the hash value and (ii) a ciphertext of a responder authentication value, and wherein the
10 ciphertext is encrypted using at least the derived shared secret, wherein the DPP server sends the
responder ephemeral public key and the ciphertext to the wireless device before sending the
encrypted set of network credentials.

47. The system of any one of claims 42 to 46, wherein the DPP server uses the
responder bootstrap private key with a plurality of different initiator bootstrap public keys,
15 wherein the plurality of different initiator bootstrap public keys are associated with a plurality of
different initiators.

48. A wireless device for supporting a device provisioning protocol (DPP), the wireless
device comprising:

a WAN radio for establishing a secure session with a server, and for receiving from the
20 server (i) a responder configuration, (ii) a first hash value of a responder bootstrap public key, and
for sending to the server a first ciphertext for a DPP authentication request message from an
initiator;

a WiFi radio for operating with the responder configuration, for receiving from the initiator
the DPP authentication request message with (i) the first ciphertext, (ii) the first hash value, and
25 (iii) an initiator ephemeral public key, and for transmitting to the initiator a DPP authentication
response message with a second ciphertext and a responder ephemeral public key;

a nonvolatile memory for storing (i) a responder network table with at least the first hash value and an identity for the server, and (ii) a DPP application for conducting the DPP over the WiFi radio using the responder configuration;

a processor for selecting the identity from the responder network table using the first hash value, for validating the initiator ephemeral public key, for encrypting the initiator ephemeral public key and the first ciphertext for the secure session, and for storing in volatile memory a third ciphertext from the server, wherein the third ciphertext includes data for a DPP authentication confirm message; and

a data bus for electrically connecting the WAN radio, WiFi radio, nonvolatile memory, and processor, and for transferring from the WAN radio to the WiFi radio a fourth ciphertext of network credentials for the initiator.

49. The wireless device of claim 48, wherein the wireless device (i) does not store and (ii) does not receive a responder bootstrap private key corresponding to the responder bootstrap public key.

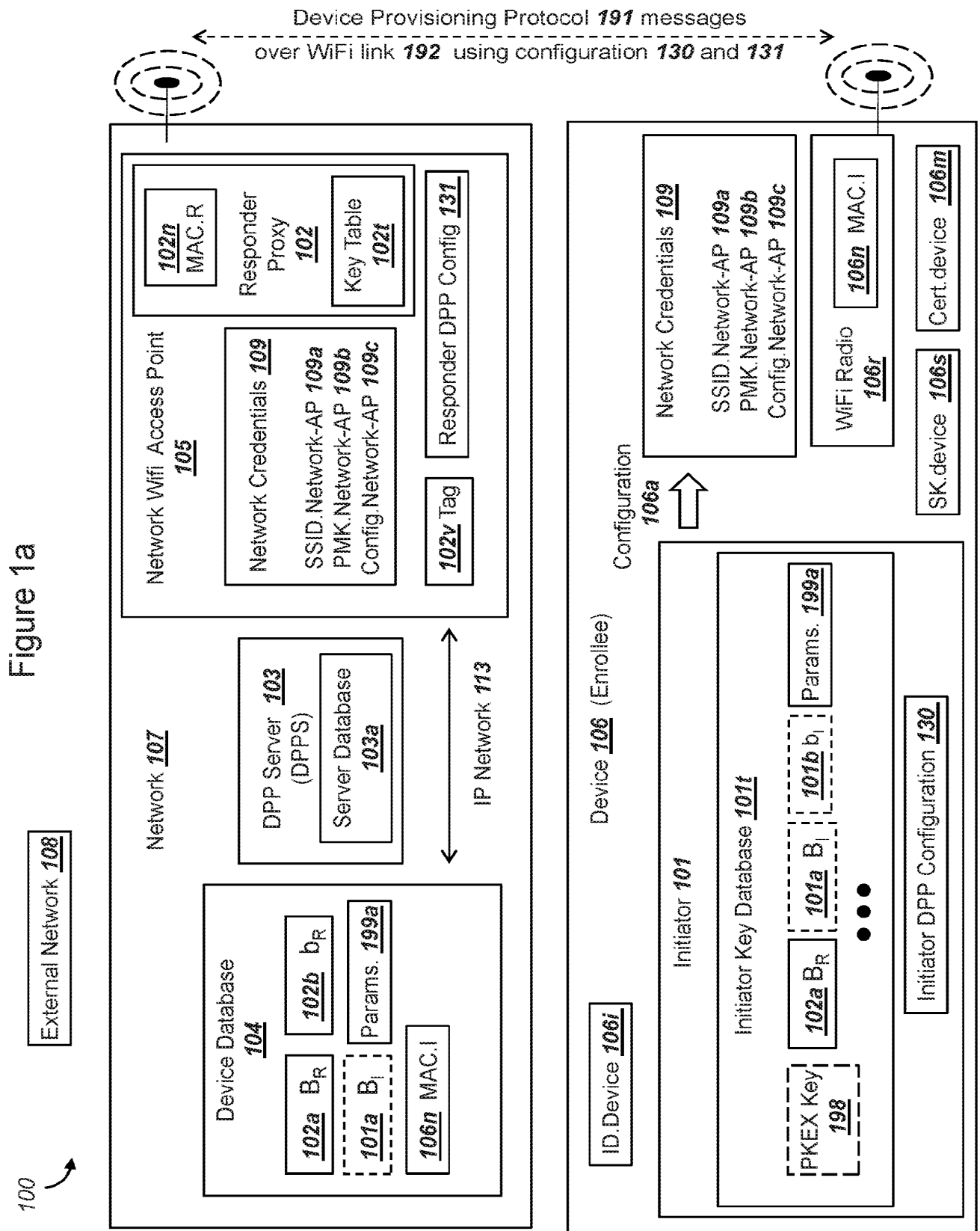
50. The wireless device of claim 48 or 49, wherein the wireless device comprises a responder for the DPP, and wherein the DPP authentication request message is received by the wireless device as a broadcast message over a WiFi link.

51. The wireless device of any one of claims 48 to 50, wherein the server records a responder bootstrap private key corresponding to the responder bootstrap public key, wherein the WiFi radio receives a second hash value of an initiator bootstrap public key with the DPP authentication request, and wherein the WAN radio sends the second hash value to the server.

52. The wireless device of any one of claims 48 to 51, wherein the third ciphertext is encrypted with a symmetric ciphering key, and wherein the symmetric ciphering key is derived by the server using an elliptic curve Diffie Hellman (ECDH) key exchange and at least (i) the responder bootstrap private key, and (ii) the initiator ephemeral public key.

53. The wireless device of claim 52, wherein the symmetric ciphering key is derived using an initiator bootstrap public key.

54. The wireless device of any one of claims 48 to 53, wherein the WiFi radio transmits the DPP authentication confirm message that includes the third ciphertext, wherein the WiFi radio
5 receives a DPP configuration request message after the WiFi radio transmits the DPP authentication confirm message, wherein the WiFi radio transmits a DPP configuration response message with the fourth ciphertext to the initiator after the WiFi radio receives the DPP configuration request message.



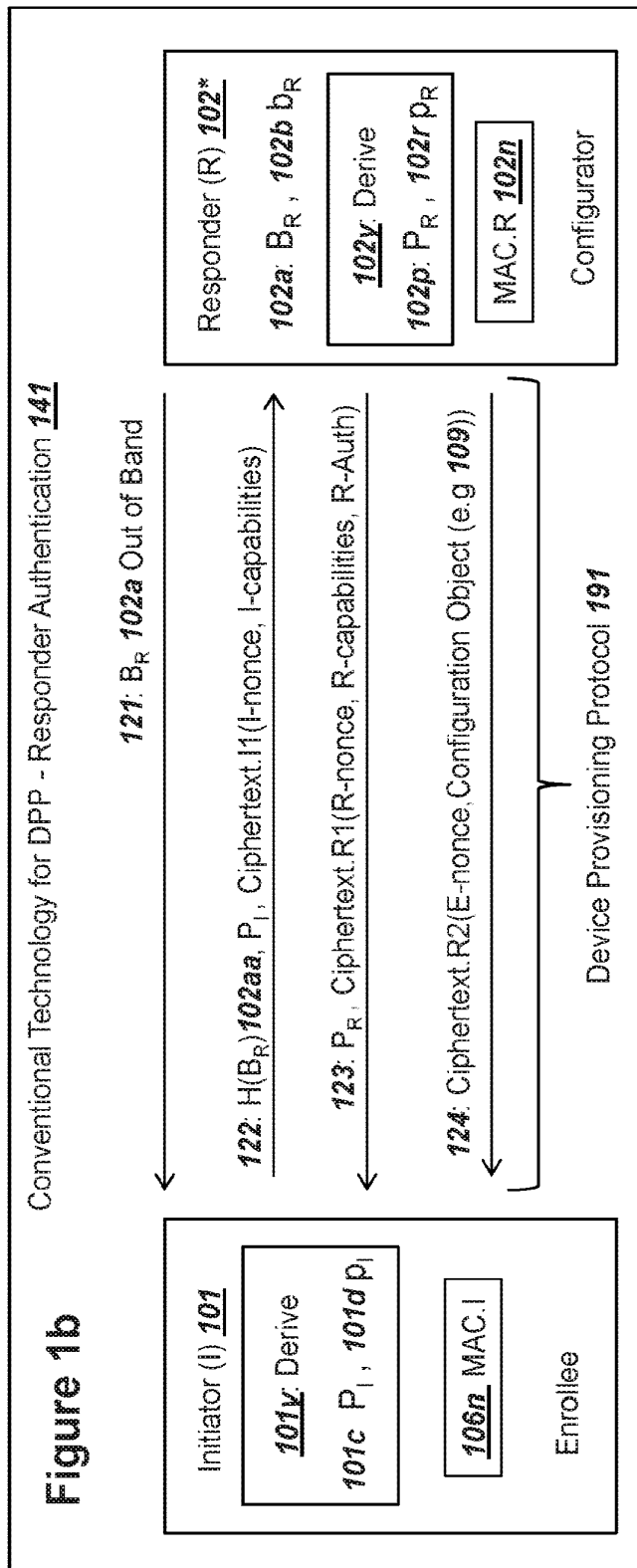
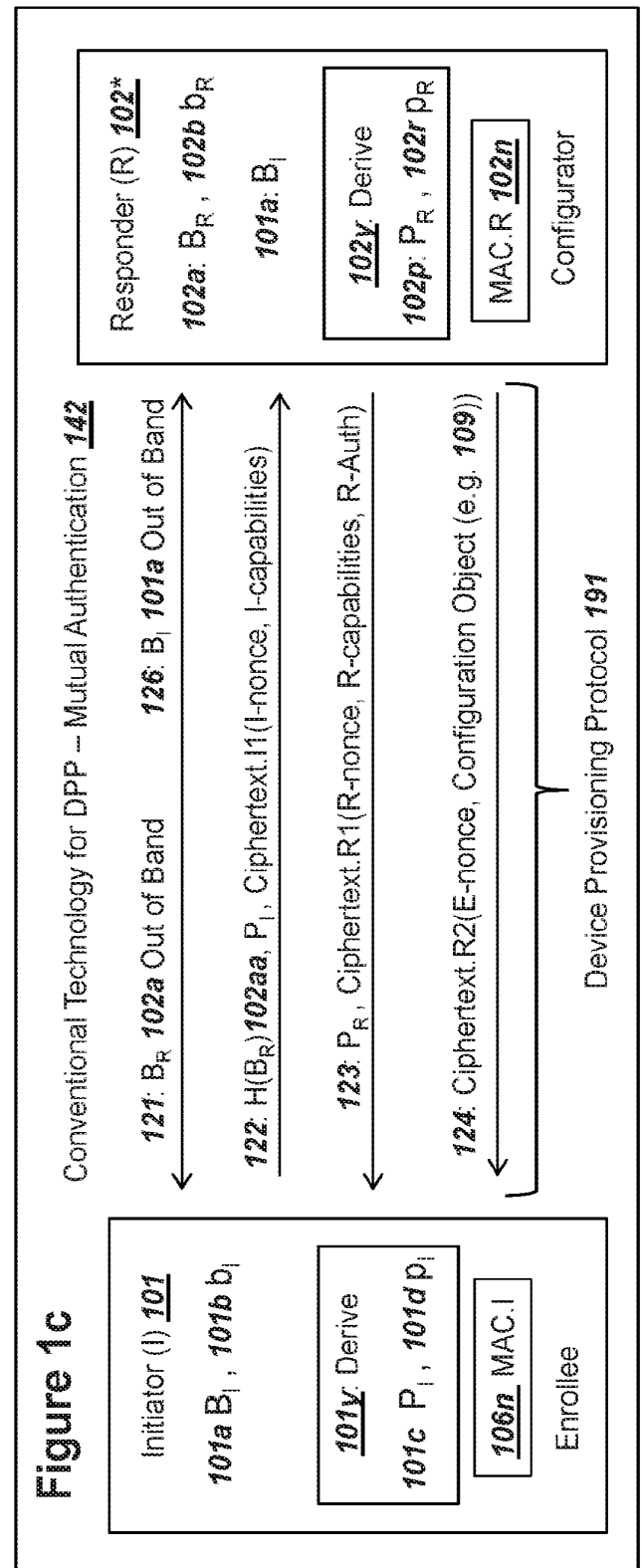
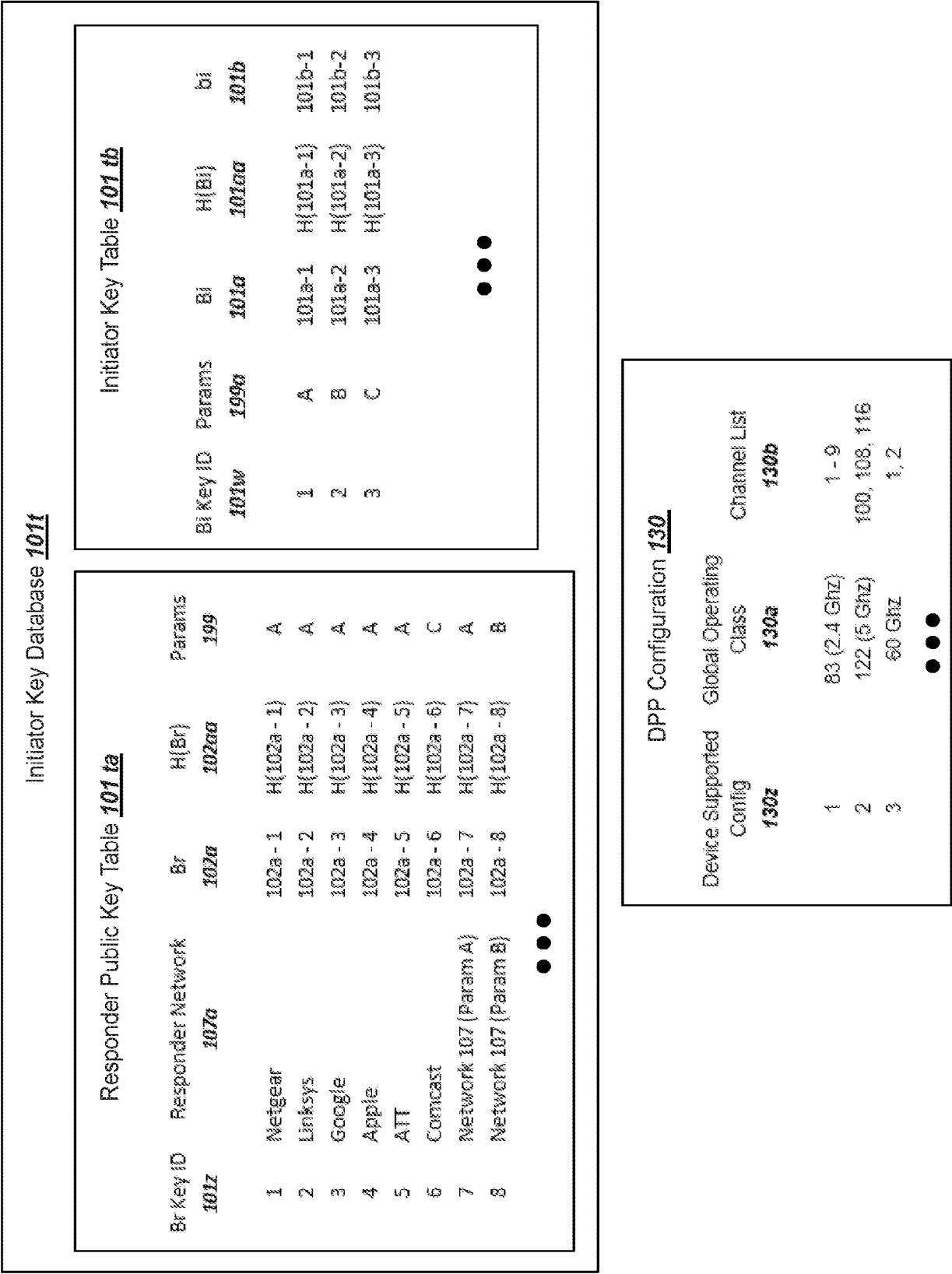


Figure 1d



DPP Configuration **130**

Device Supported Config	Global Operating Class	Channel List
130z	130a	130b
1	83 (2.4 Ghz)	1-9
2	122 (5 Ghz)	100, 108, 116
3	60 Ghz	1, 2
...		

Figure 1e

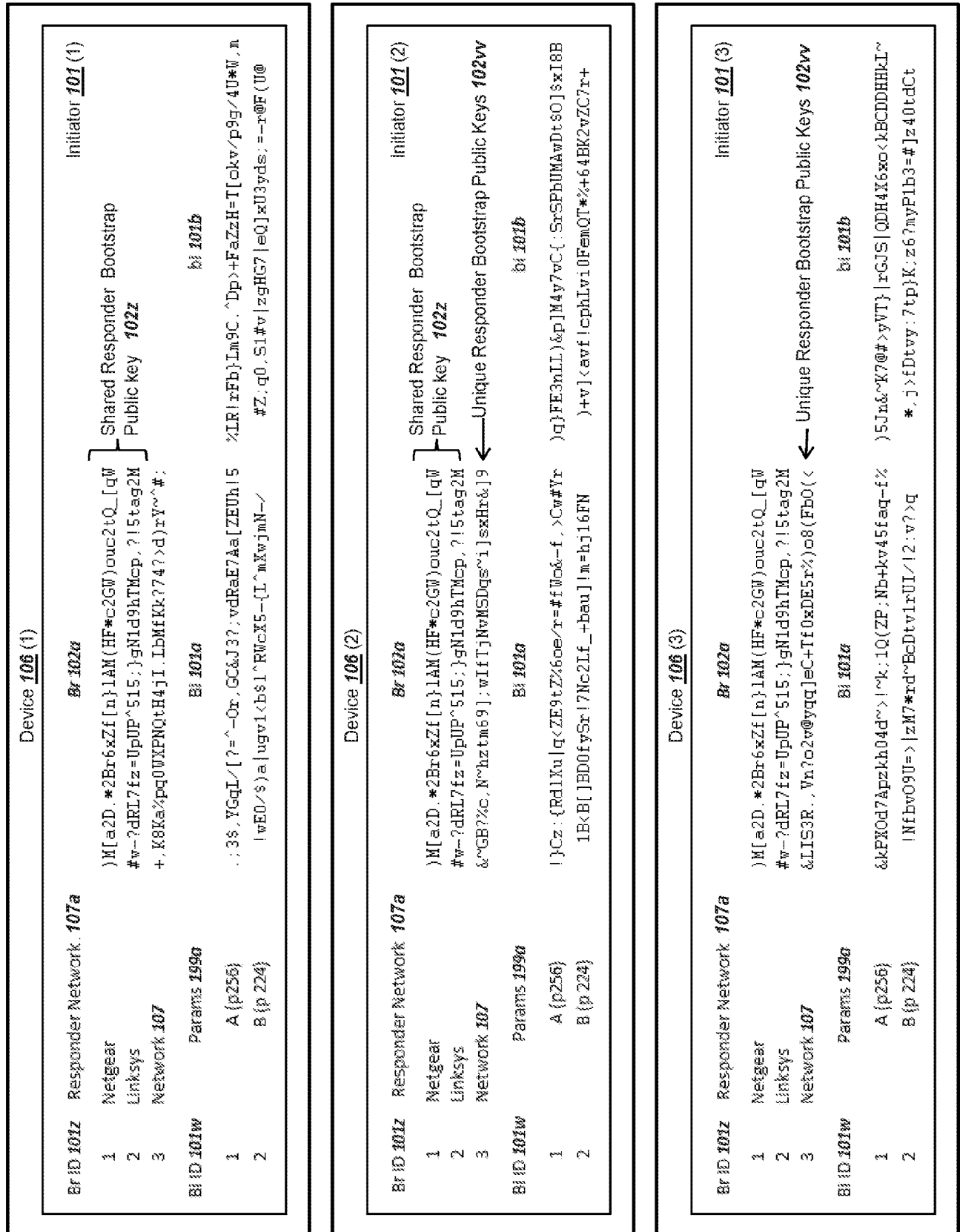


Figure 1f

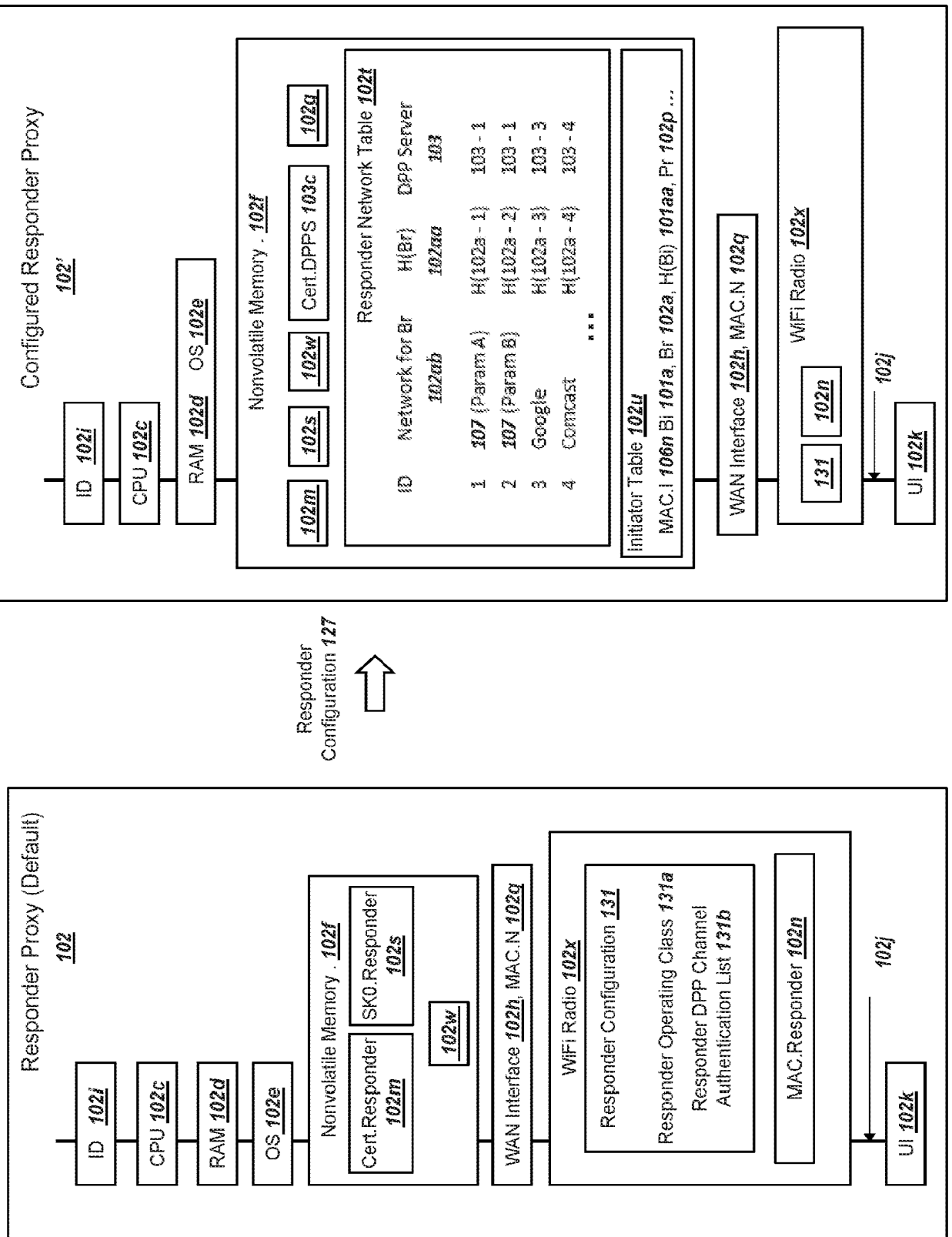


Figure 2a

Device Database <u>104</u>					
Index	Device Identity	MAC	Initiator	Responder Identity	Responder Mode
1	20001	38607720A	6cf049080	6cf049080	Fig. 3
2	20002	386077209	6cf049080	6cf049080	Fig. 5a
3a	20003	386077400	6cf049080	6cf049080	102*
3b	20003	386077400	6cf049080	6cf049080	102*
4	20004	3860773cc	1c6f654d2	1c6f654d2	Fig. 3
5a	20005	386077a09	1c6f654d2	1c6f654d2	Fig. 5b
5d	20006	386077a09	1c6f654d2	1c6f654d2	Fig. 5

Responder				Initiator				Key Table <u>104k</u>	
Bootstrap	Public Key (Br)	Private Key (br)	Responder Bootstrap	Public Key (Bi)	Private Key (br)	Responder Bootstrap	Public Key (Bi)	Private Key (br)	Initiator Bootstrap
102a	102a	102b	102a	102a	102b	102a	102a	102b	102a
Br-1 (Shared Keys)	Br-1 (Shared Keys)	br-1 (Shared Keys)	Br-1 (Shared Keys)	Br-1 (Shared Keys)	br-1 (Shared Keys)	Br-1 (Shared Keys)	Br-1 (Shared Keys)	br-1 (Shared Keys)	Br-1 (Shared Keys)
Br-1 (102z)	Br-1 (102z)	br-1 (102z)	Br-1 (102z)	Br-1 (102z)	br-1 (102z)	Br-1 (102z)	Br-1 (102z)	br-1 (102z)	Br-1 (102z)
Br-3(B)	Br-3(B)	br-3(B)	Br-3(B)	Br-3(B)	br-3(B)	Br-3(B)	Br-3(B)	br-3(B)	Br-3(B)
Br-4(C)	Br-4(C)	br-4(C)	Br-4(C)	Br-4(C)	br-4(C)	Br-4(C)	Br-4(C)	br-4(C)	Br-4(C)
Br-5(A)	Br-5(A)	br-5(A)	Br-5(A)	Br-5(A)	br-5(A)	Br-5(A)	Br-5(A)	br-5(A)	Br-5(A)
Br-5(D)	Br-5(D)	br-5(D)	Br-5(D)	Br-5(D)	br-5(D)	Br-5(D)	Br-5(D)	br-5(D)	Br-5(D)

Cryptographic Parameters <u>199</u>					
Set	Key Length	Curve	Hash	AES-SIV Key Length	Asymmetric Encryption Parameters
199a	199b	199c	199e	199f	199h
A	256	p256	SHA-256	256	Elgamal, p, g
B	384	p384	SHA-384	384	Elgamal, p, g
C	512	secp521r1	SHA-512	512	RSA
D	384	Curve25519	SHA-3	384	Elgamal, p, g

7/20

Figure 2b

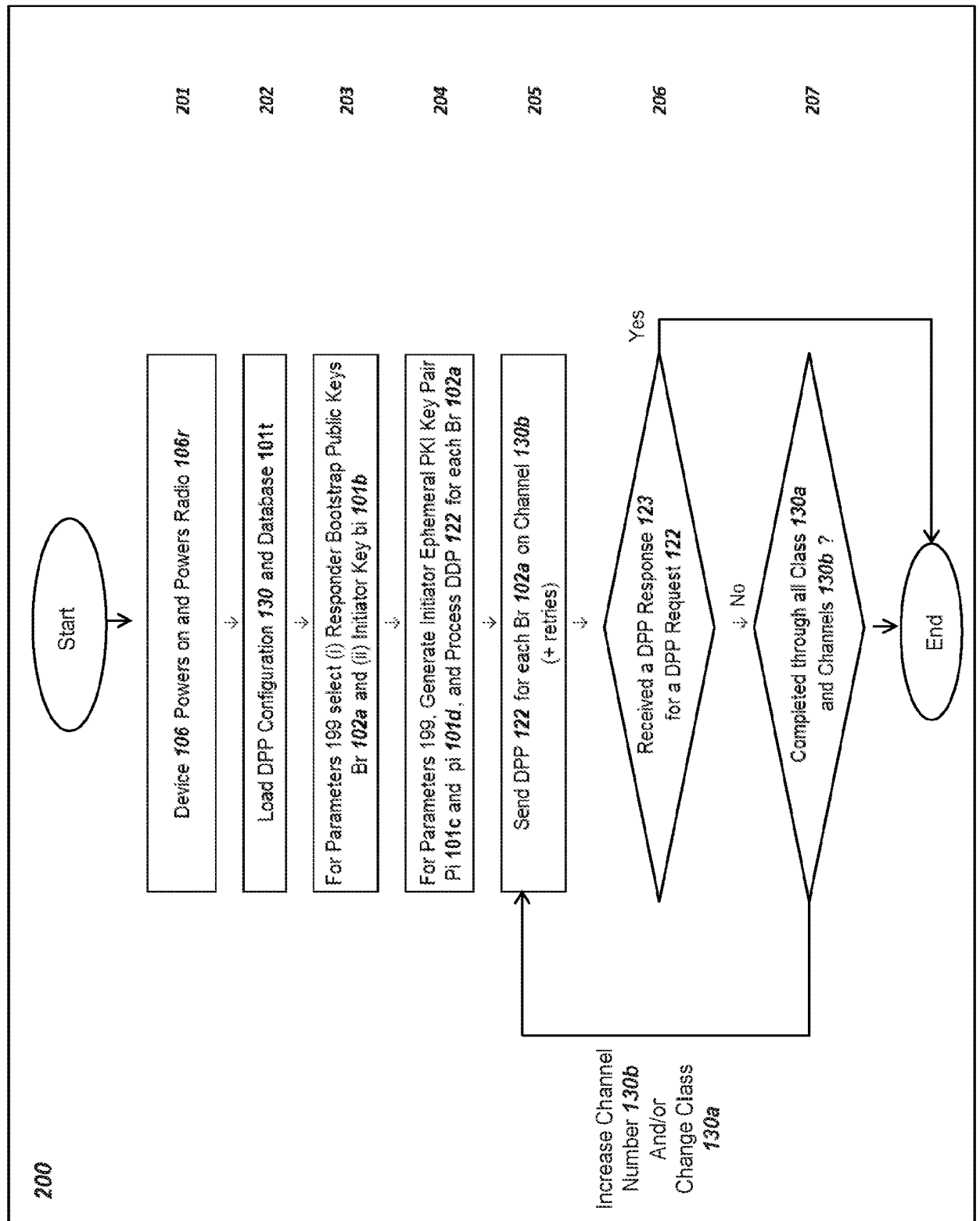
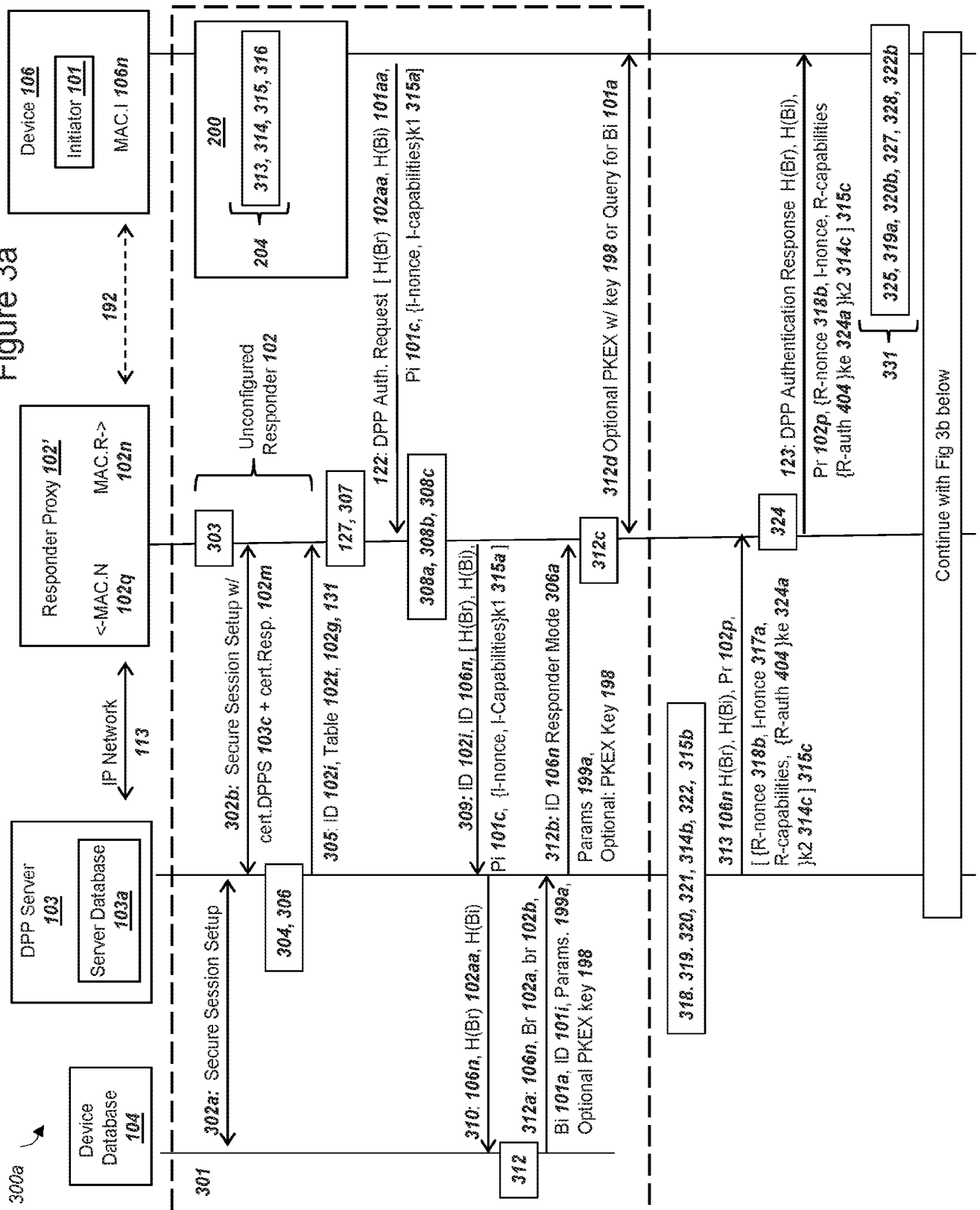


Figure 3a



9/20

Figure 3b

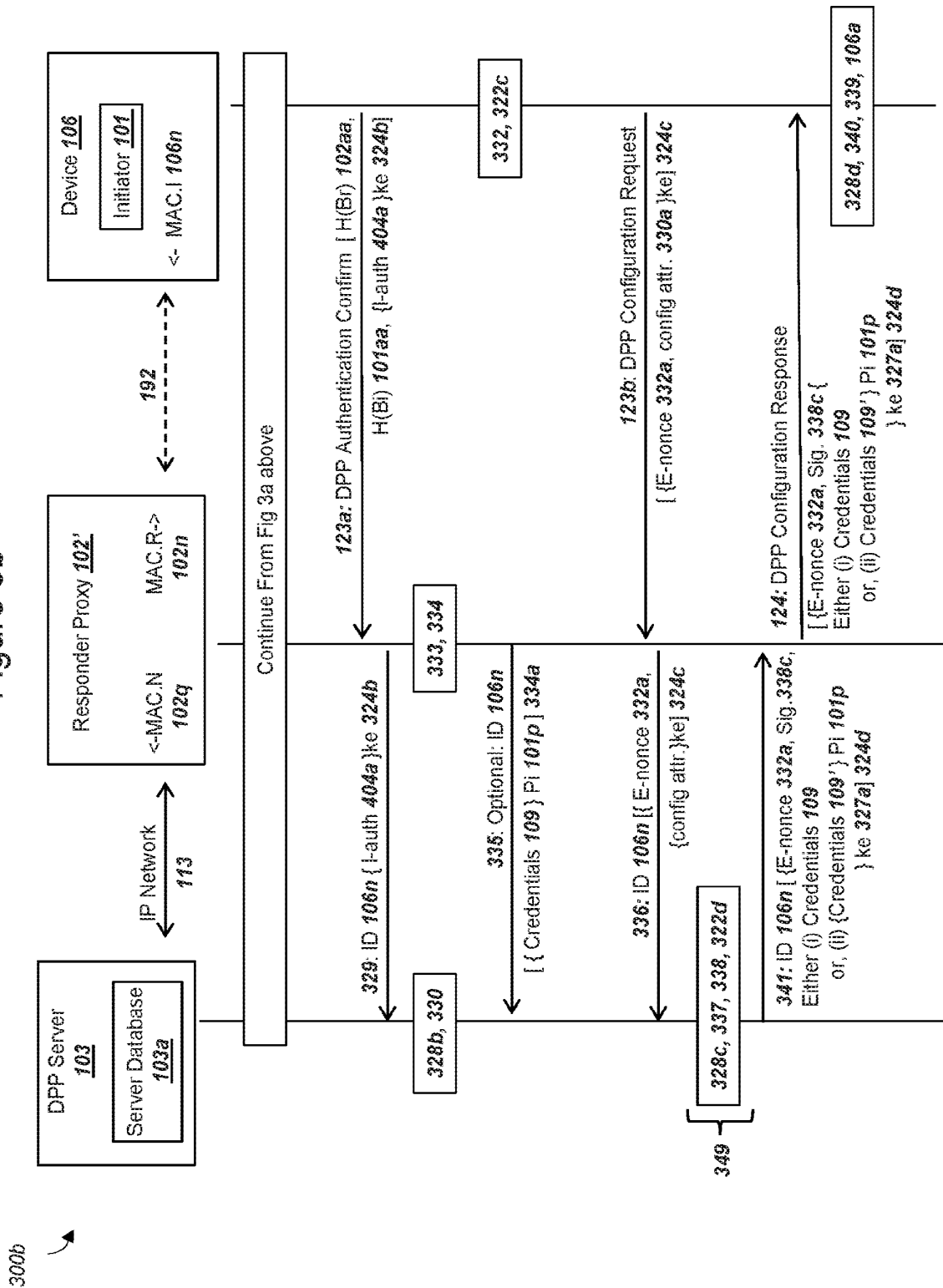


Figure 3c

Server Database <u>103a</u>												
MAC Initiator <u>106n</u>	Responder MAC <u>102n</u>	Responder Mode <u>306a</u>	Responder Cryptog. Params <u>199a</u>	Responder		Initiator		PKEX		Shared		Shared Secret ke <u>327a</u>
				Ephem. Public Key <u>102p</u>	Ephem. Private Key <u>102r</u>	Ephem. Public Key <u>101c</u>	Ephem. Private Key <u>101r</u>	Key <u>198</u>	Secret k1 <u>314a</u>	Secret k2 <u>314c</u>	Secret ke <u>327a</u>	
38607720A	6cf049080	Fig. 3	A	Pr-1	pr-1	Pl-1	Pl-1	X1	k1-1	k2-1	ke-1	
386077208	4a508b99	Fig. 5a	A	-	-	Pl-2	Pl-2	-	-	-	-	
386077209	1c61654d2	Fig. 5b	A	-	pr-2	Pl-2	Pl-2	NA	-	-	-	
386077400	7071bce29	102*	B	-	-	-	-	NA	-	-	-	
3860773cc	7071bce29	Fig. 3	C	Pr-4	pr-4	Pl-4	Pl-4	X4	k1-4	k2-4	ke-4	

• • •

Device Credentials
Table 345

Device Identity <u>106i</u>	MAC. Initiator <u>106n</u>	Primary		Backup	
		Network Credentials <u>350a</u>	Network Credentials <u>350b</u>	Network Credentials <u>350c</u>	Network Credentials <u>350d</u>
20001	38607720A	109 - 1	109 - 1	109 - 3	109 - 3
20002	386077209	109 - 1	109 - 1	109 - 4	109 - 4
20003	386077400	109' - 1	109' - 1	109' - 3	109' - 3
20004	3860773cc	109' - 2	109' - 2	109' - 4	109' - 4
20005	386077a09	109 - 2	109 - 2	109 - 5	109 - 5

11/20

Figure 3e

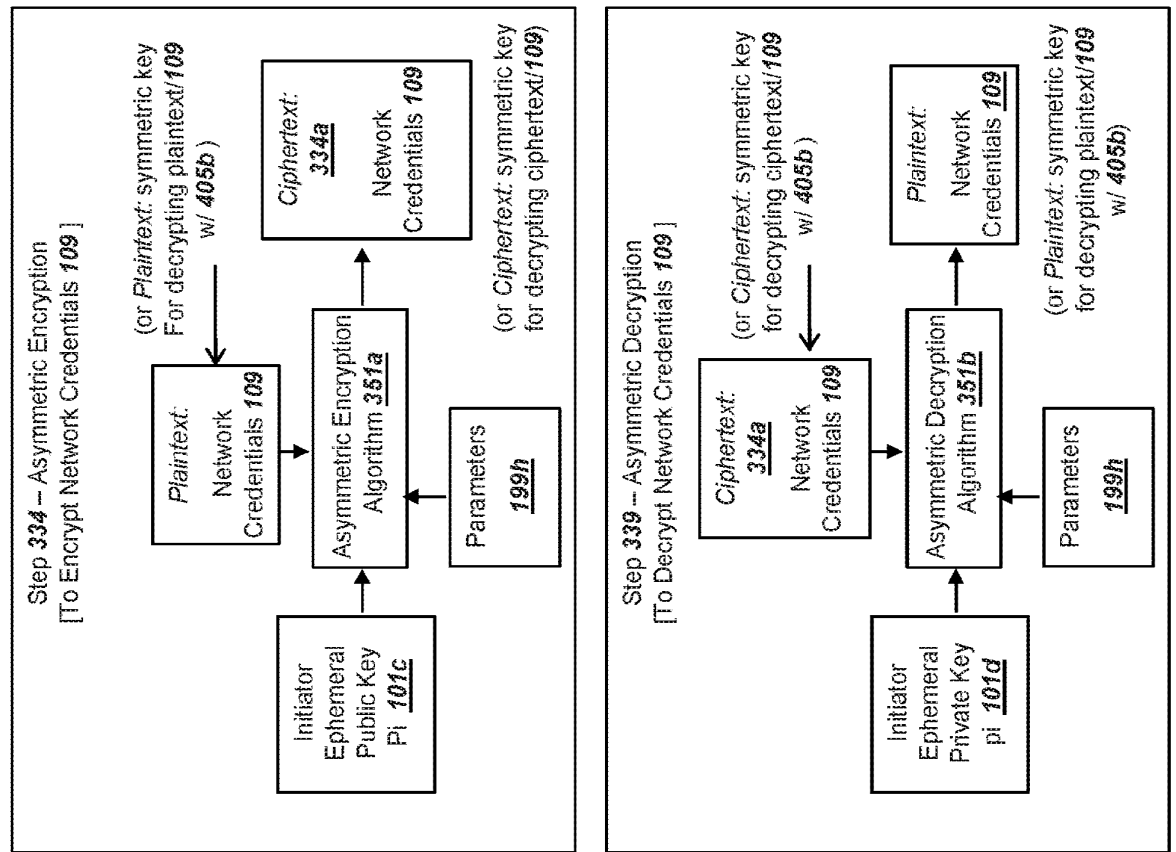
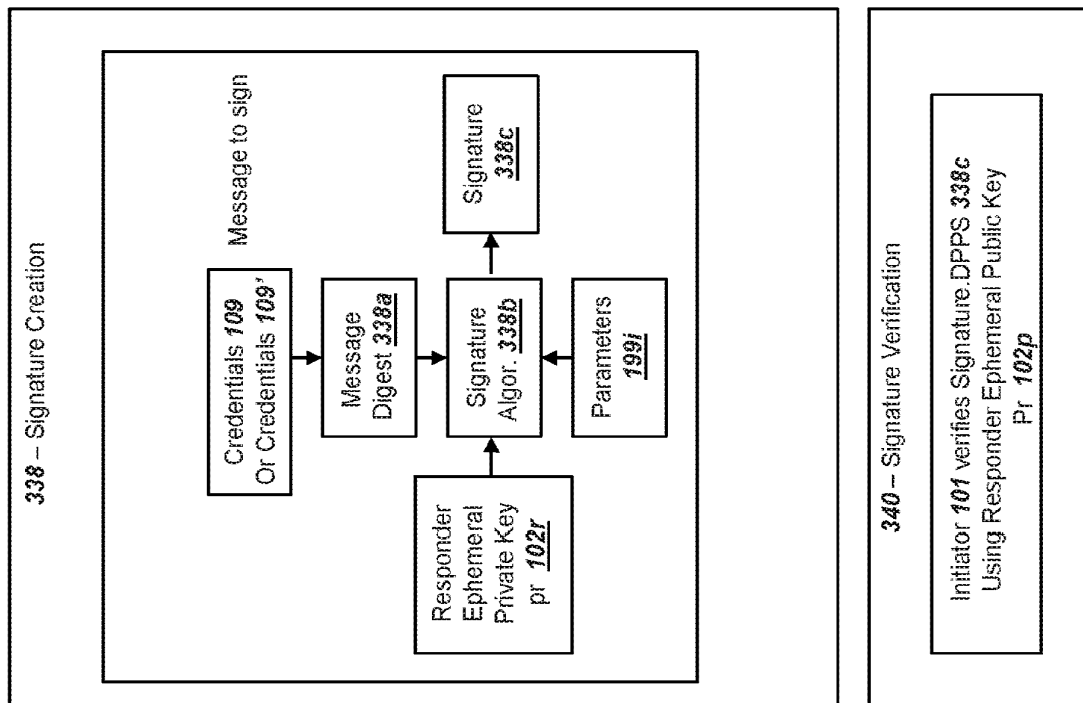
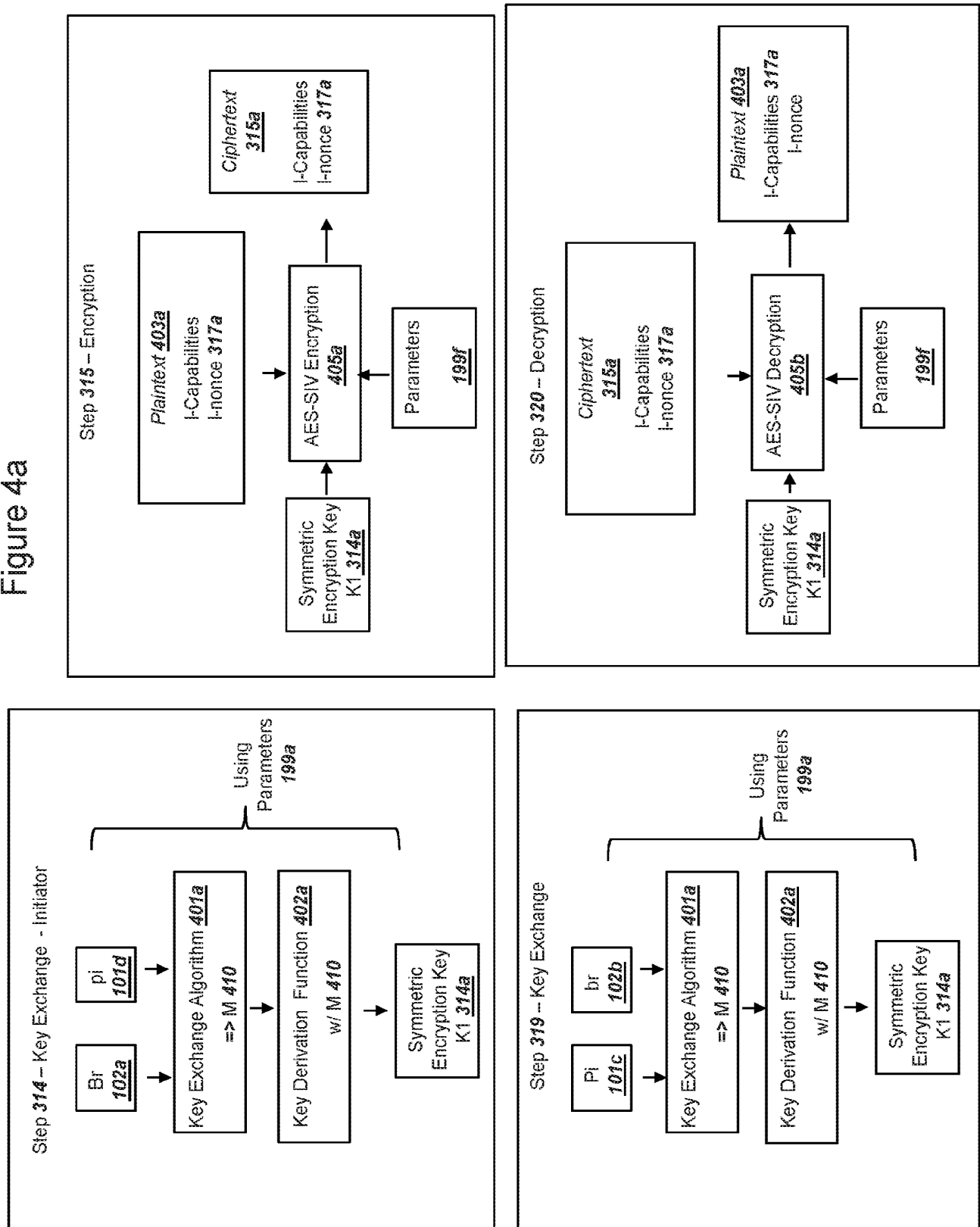


Figure 3d



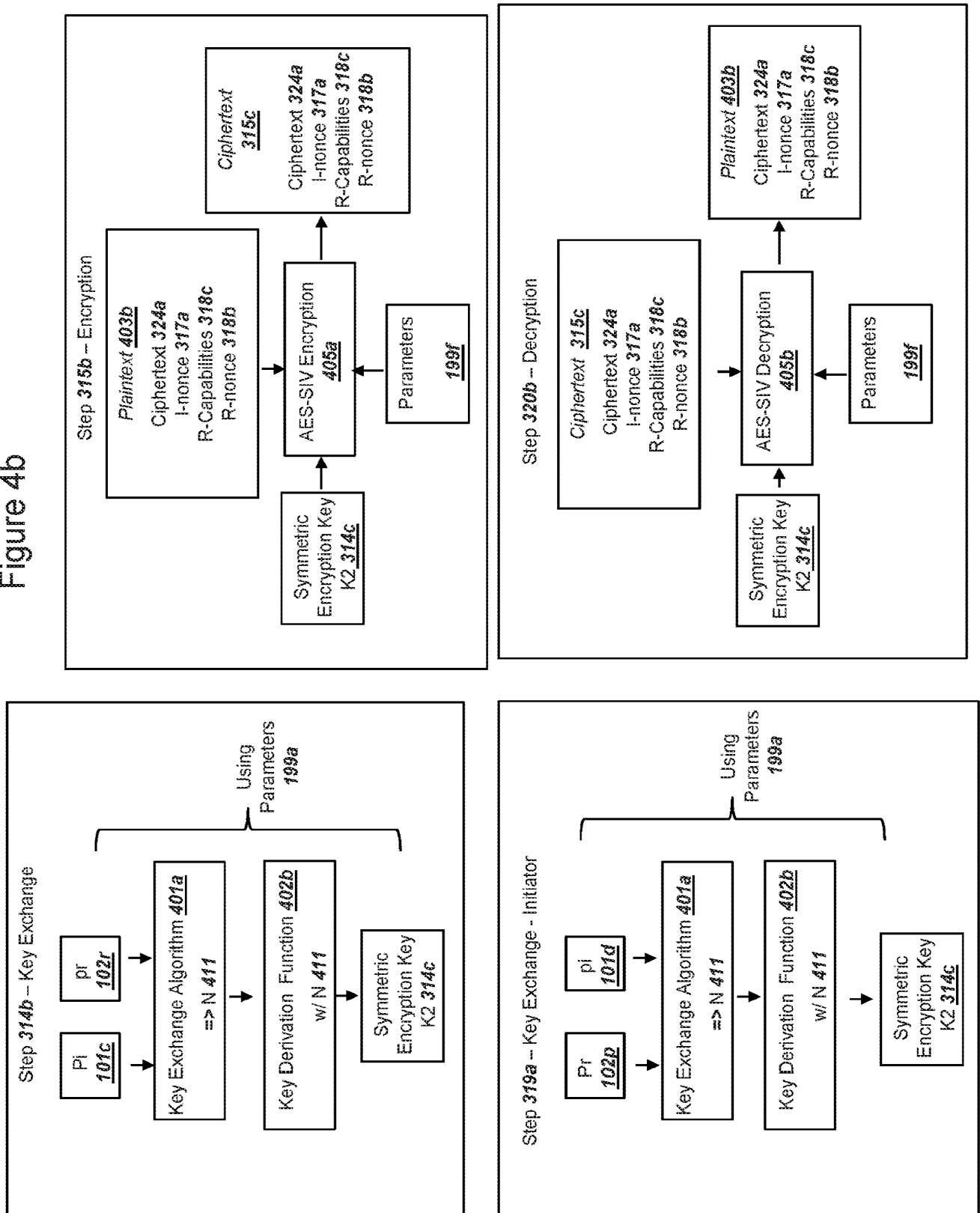
12/20

Figure 4a



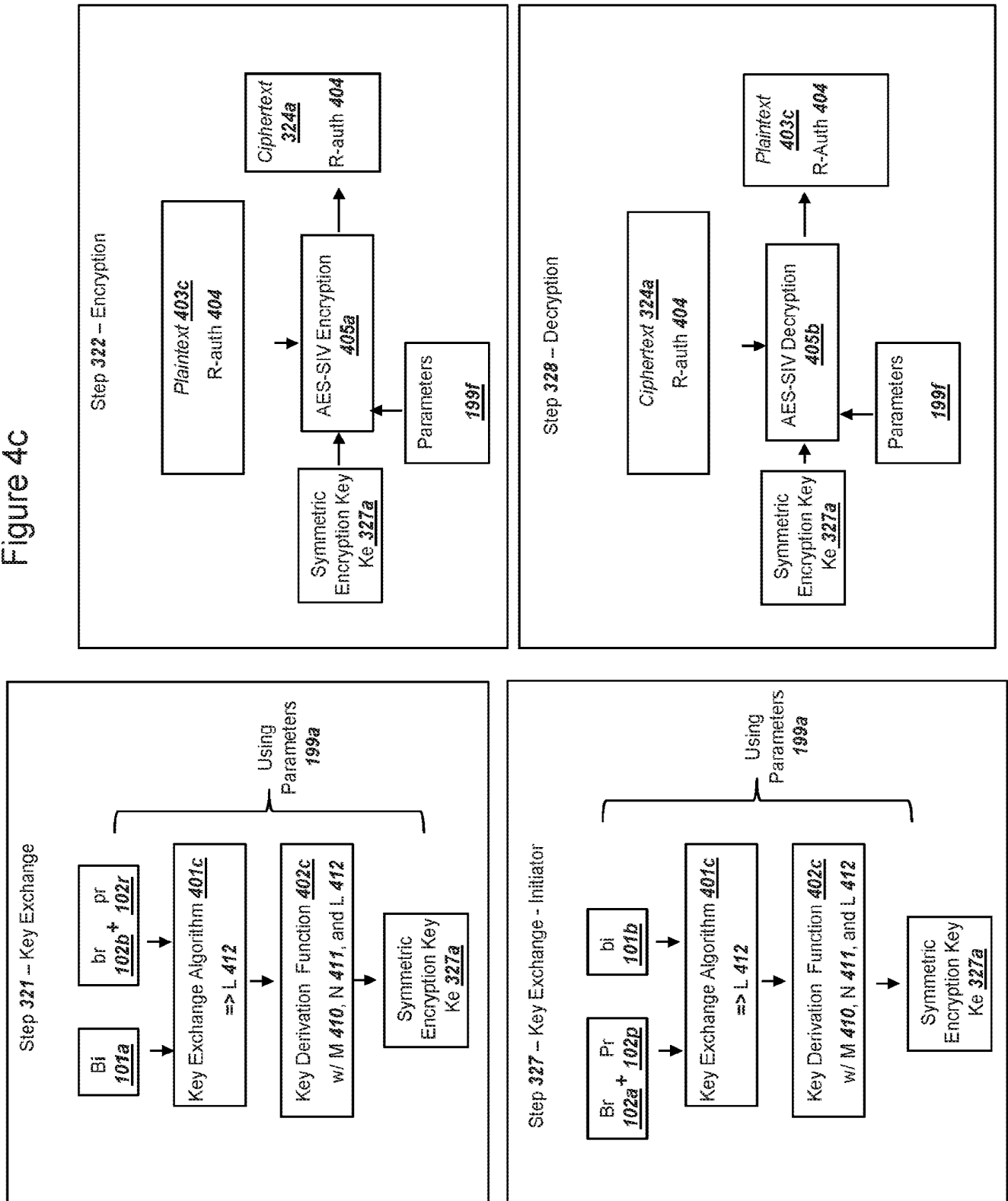
13/20

Figure 4b



14/20

Figure 4c



15/20

Figure 4d

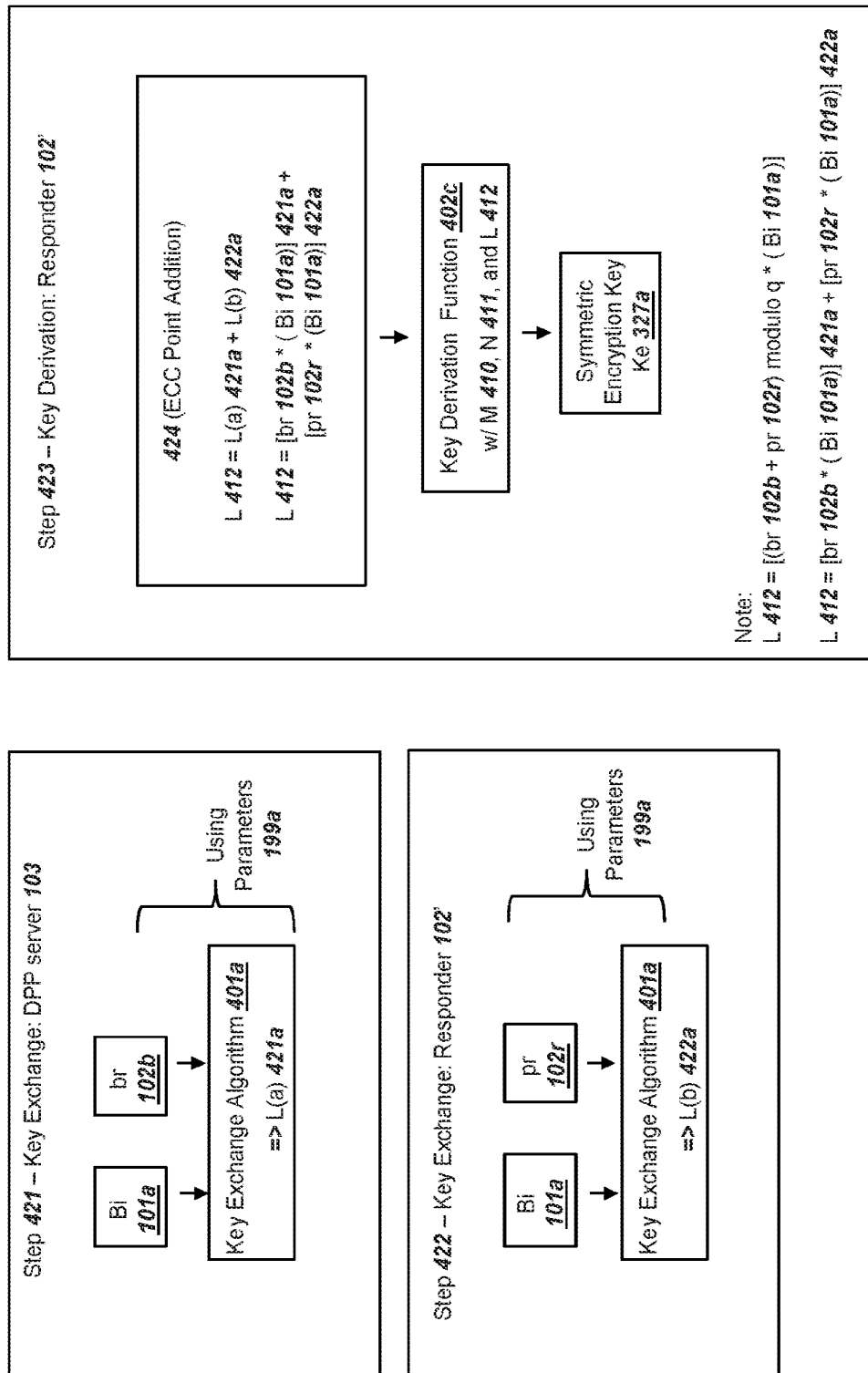
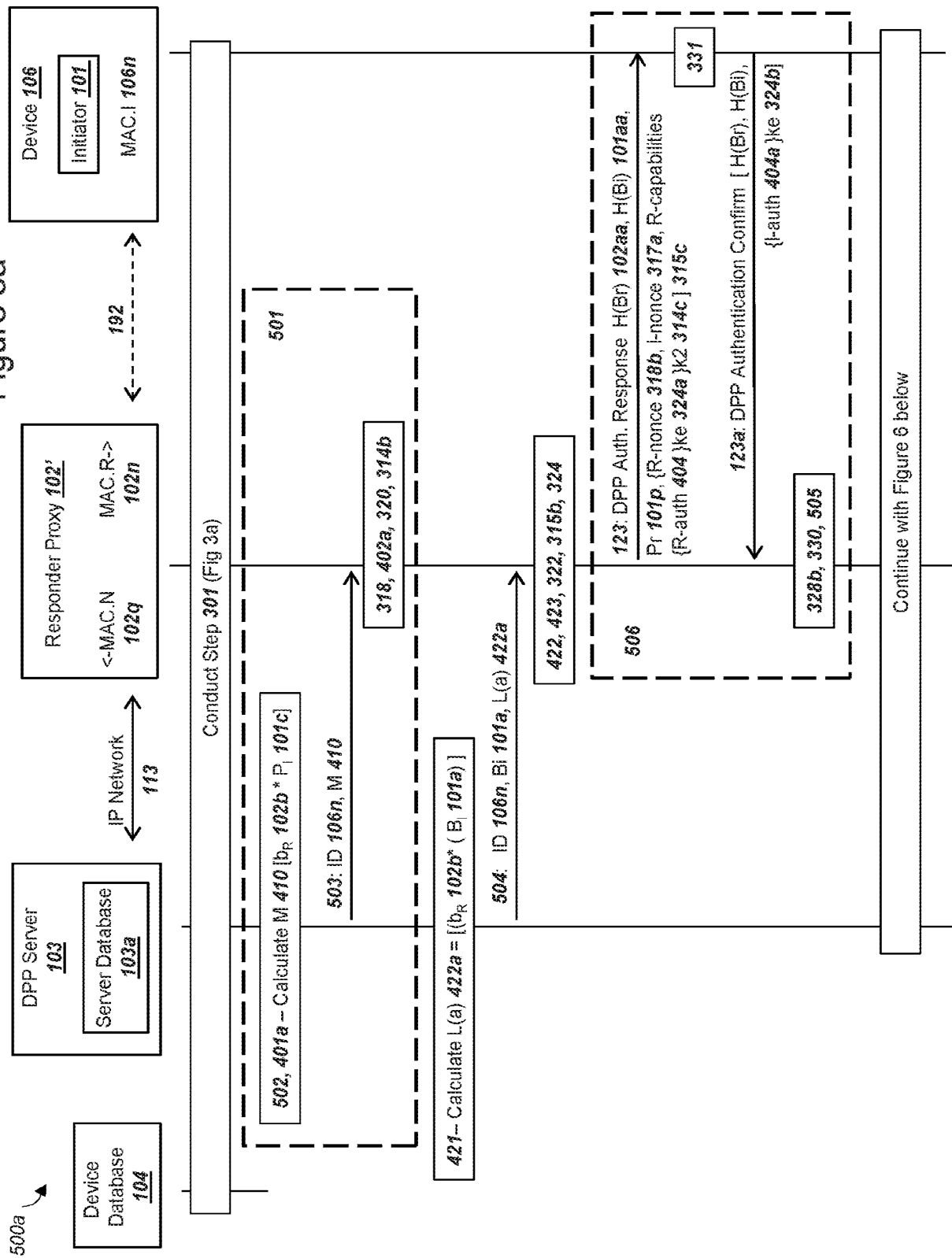


Figure 5a



17/20

Figure 5b

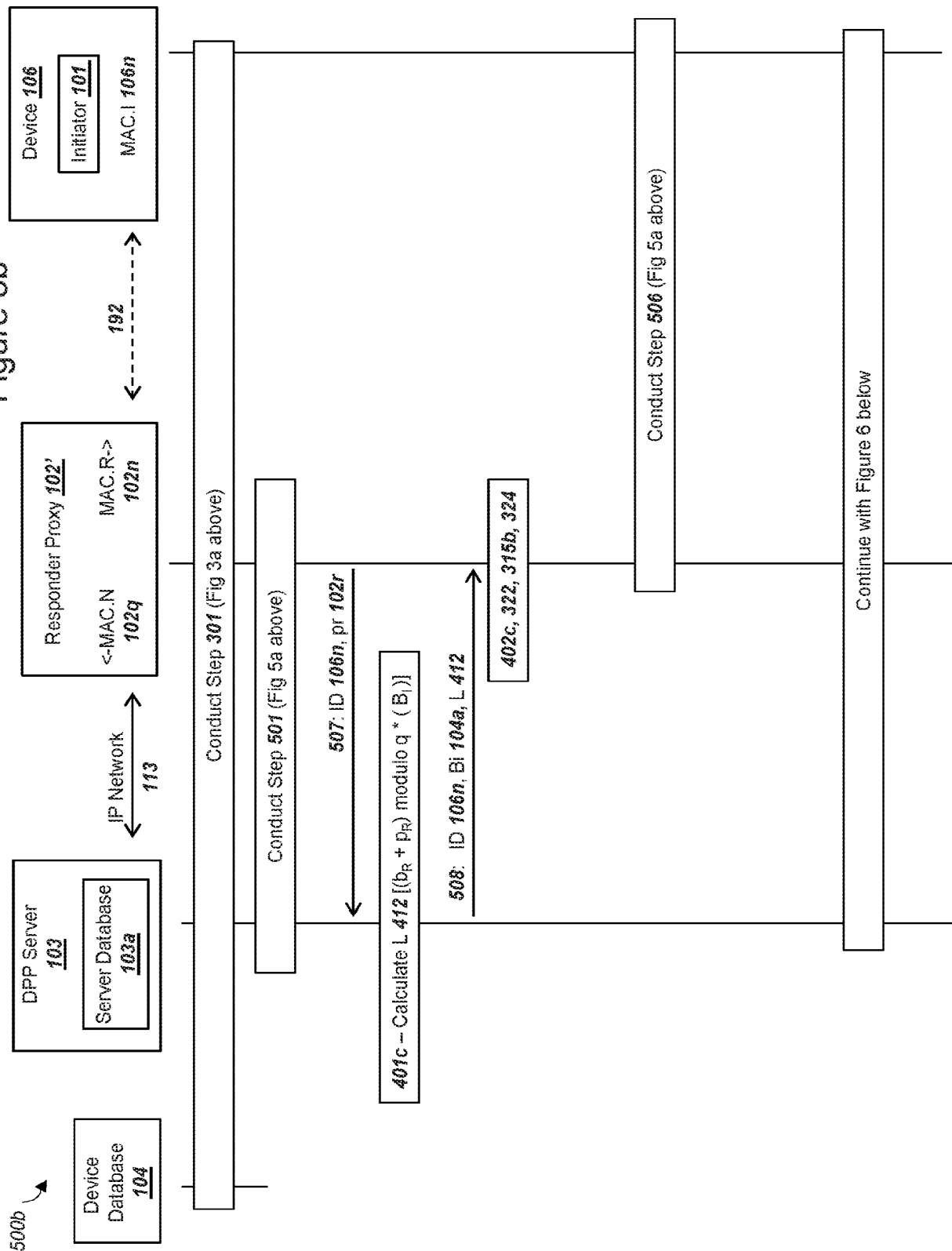


Figure 6

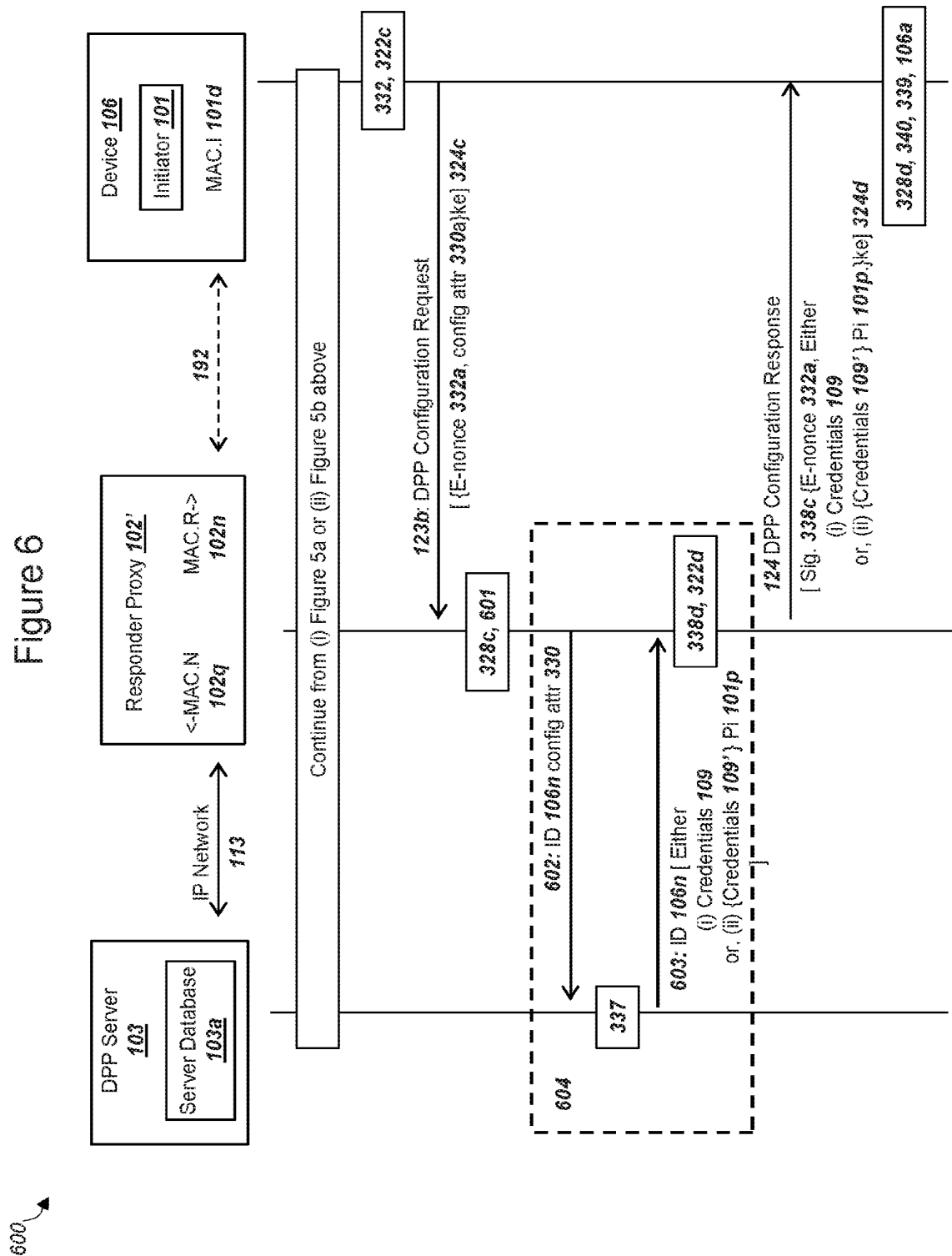


Figure 7

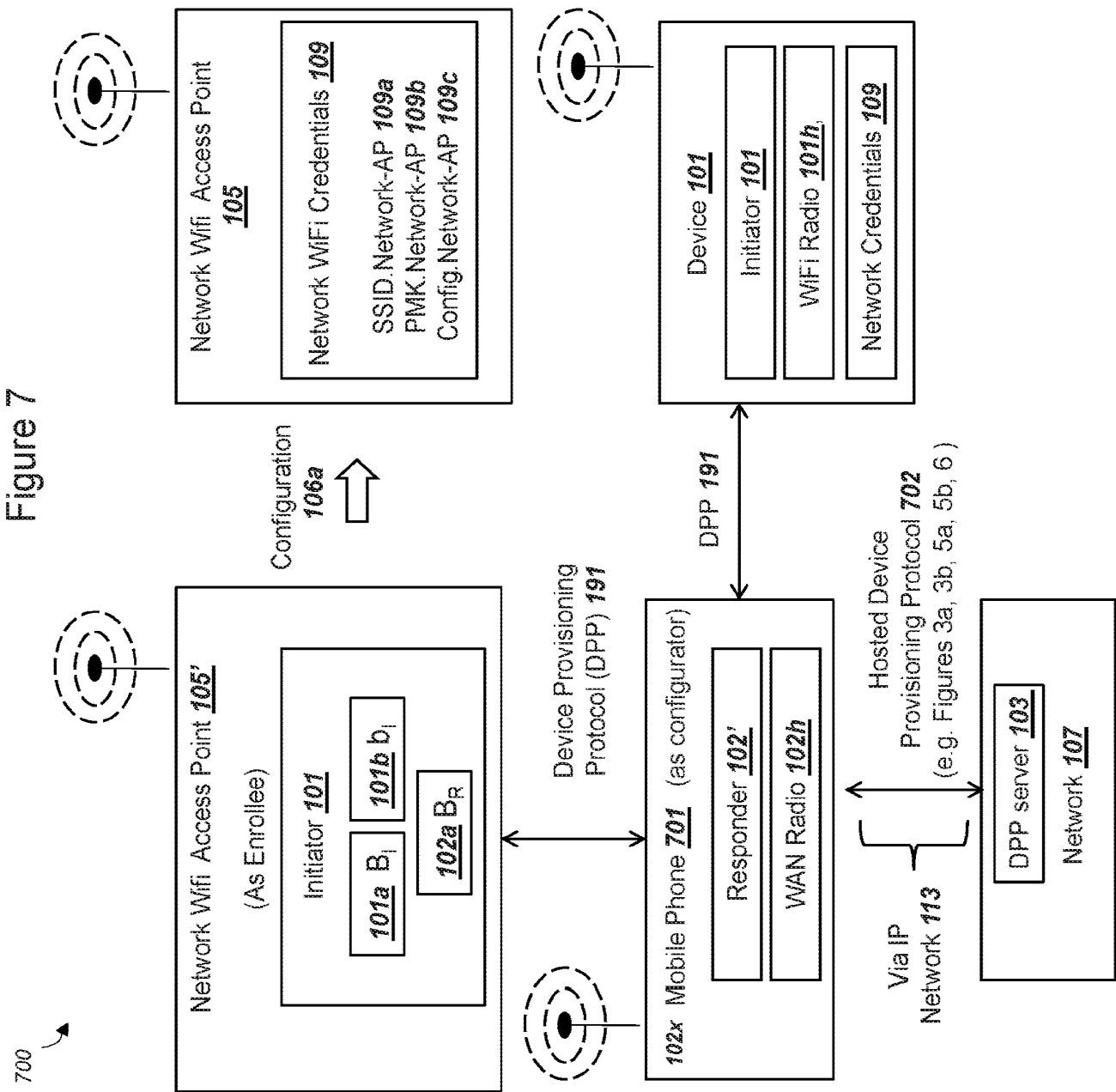


Figure 8

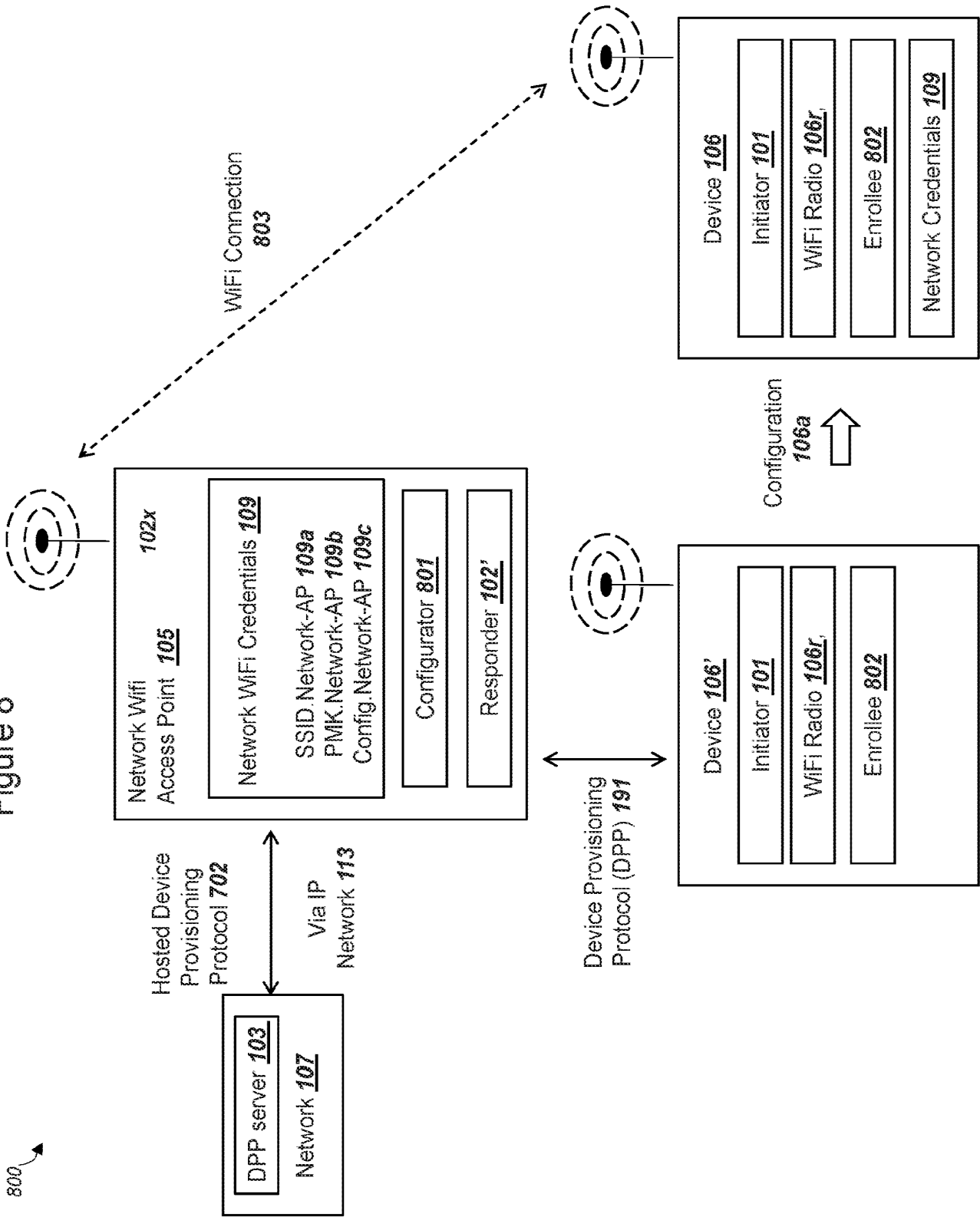
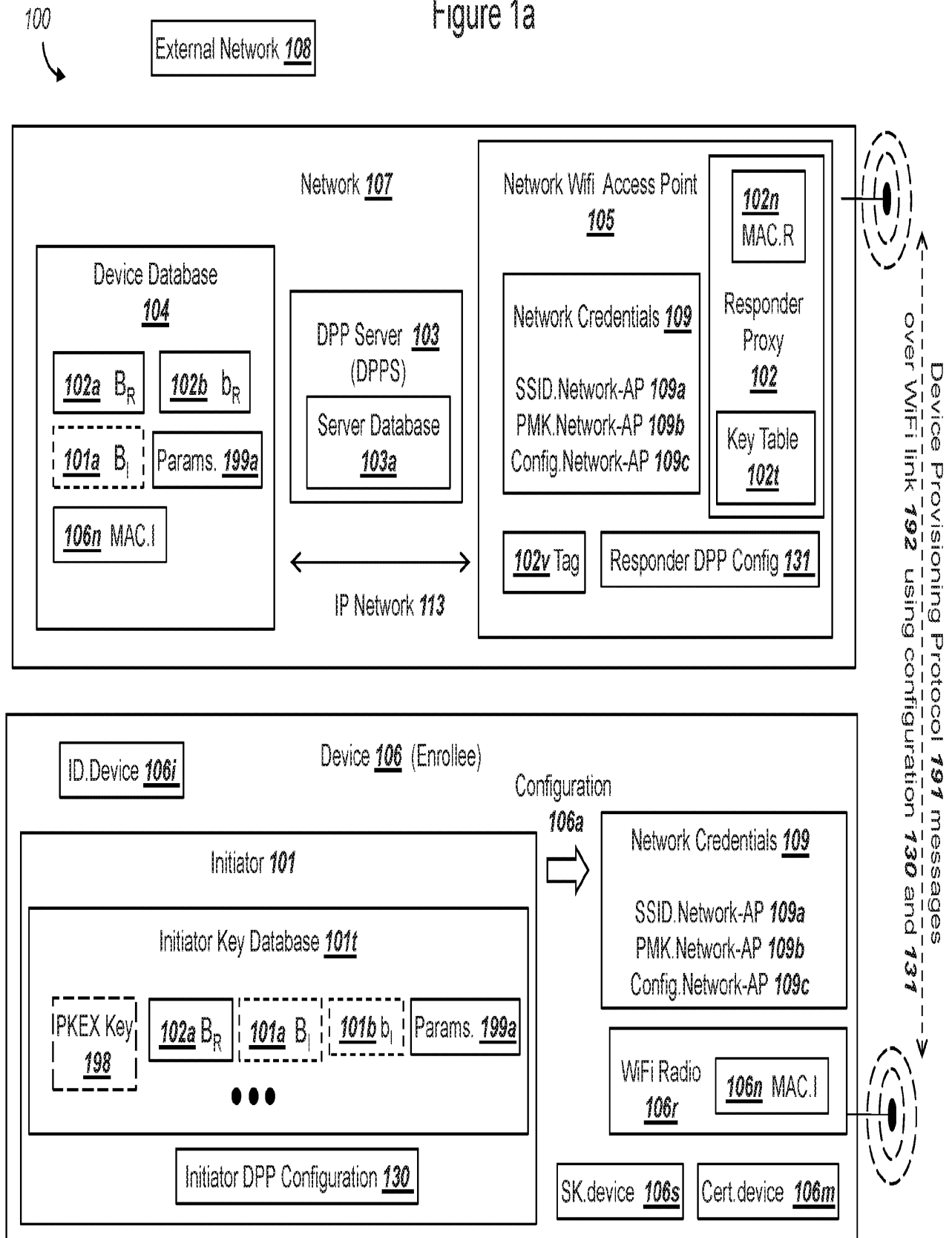


Figure 1a





(51) International Patent Classification:

H04W 4/70 (2018.01) *H04L* 9/28 (2006.01)

Published:

— with international search report (Art. 21(3))

(21) International Application Number:

PCT/US2019/051481

(22) International Filing Date:

17 September 2019 (17.09.2019)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

62/732,930 18 September 2018 (18.09.2018) US

(71) Applicant: **IOT AND M2M TECHNOLOGIES, LLC**
[US/US]; 807 Davis Street, Unit 2207, Evanston, IL
60201-7104 (US).

(72) Inventor: **NIX, John, A.**; C/o IoT and M2M Technolo-
gies, LLC, 807 Davis Street, Unit 2207, Evanston, IL
60201-7104 (US).

(74) Agent: **THOMSON, Kirsten, L.**; McDonnell Boehnen
Hulbert & Berghoff LLP, 300 South Wacker Drive, Chica-
go, IL 60606 (US).

(81) Designated States (*unless otherwise indicated, for every
kind of national protection available*): AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ,
CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO,
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,
HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP,
KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME,
MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ,
OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA,
SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN,
TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (*unless otherwise indicated, for every
kind of regional protection available*): ARIPO (BW, GH,
GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ,
UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ,
TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK,
EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV,
MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,
TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW,
KM, ML, MR, NE, SN, TD, TG).

(54) Title: STORAGE MEMORY UNIT WITH A SHARED NONVOLATILE MEMORY INTERFACE FOR A RADIO

(57) Abstract: A storage radio unit (SRU) for a device can include a radio, embedded universal integrated circuit card (eUICC), a processor, an antenna, and nonvolatile memory. The SRU can support Micro SD card form factor standards and record a file system for a device. The device can be associated with a service provider and the SRU can be associated with a network provider. The radio can support Narrowband Internet of Things (NB-IoT) standards. The SRU can operate a file system interface (FSI) for the radio, where the device records application data in a file of the FSI. The SRU can attach to a wireless NB-IoT network using credentials recorded in the eUICC. The SRU can read the file of the FSI, and compress, encrypt, and transmit the application data to a network provider via the radio. The network provider can transmit the application data via TLS to the service provider.



WO 2020/061023 A1

STORAGE MEMORY UNIT WITH A SHARED NONVOLATILE MEMORY INTERFACE FOR A RADIO

BACKGROUND

Technical Field

The present methods and systems relate to enabling communications between a device and a wireless network, and more particularly, to operating a storage radio unit (SRU) in the device, where the SRU operates (i) nonvolatile memory, (ii) a shared nonvolatile memory interface with the device, and (iii) a radio unit for transmitting application data from the device to the wireless network using the shared nonvolatile memory interface.

Description of Related Art

The ability to connect transducers such as sensors and actuators with a network is a growing field with many economical applications. As the cost for both electronic hardware and bandwidth continues to decrease, the use of networked transducers is expected to continue increasing over the coming decades. Connecting transducers to a network can be referred to as “machine-to-machine (M2M) communications” or “the Internet of Things (IoT).” Among many potential benefits, IoT technologies allow automated monitoring and/or control of assets, equipment, personnel, or a physical location where manual monitoring may not be economical. Many applications for the “Internet of Things” significantly reduce costs by using automated monitoring instead of manual techniques. Prominent examples of IoT applications today include monitoring and control for building heating/air-conditioning, automobiles, alarm systems, and tracking devices. Fast growing markets for IoT applications today include health applications such as the remote monitoring of a person’s fitness activity, heartbeat, or glucose levels, monitoring of industrial equipment deployed remotely in the field, and also security systems.

IoT communications can provide remote control over actuators that may be connected to a device, such as turning on or off a power switch, locking or unlocking a door, adjusting a speed of a motor, or similar remote control. A decision to change or adjust an actuator associated with a connected device can utilize one or a series of sensor measurements. As one example, an IoT device for a truck can periodically report engine status to a remote server, and if the engine is operating outside specifications such as being too hot, including potentially an “alarm” condition, then temperature and an alarm code can be reported to a central server for the

IoT device. The server can subsequently instruct the driver and/or a specified mechanic to investigate the engine for potential mechanical malfunctions or other causes. The previous example is just one of many possible applications for IoT technology.

Many IoT applications can leverage wireless networking technologies, in addition to
5 wired technologies such as Ethernet. Wireless technologies such as wireless local area networks and wireless wide area networks have proliferated around the world over the past 20 years, and usage of these wireless networks is also expected to continue to grow. Wireless local area network (LAN) technologies include WiFi and wireless wide area network (WAN) technologies include 3rd Generation Partnership Project's (3GPP) 3rd Generation (3G) Universal Mobile
10 Telecommunications System (UMTS) and 4th Generation (4G) Long-term Evolution (LTE), LTE Advanced, Narrow-band Internet of Things (NB-IoT), as well as 5G networks. The many options to connect a transducer device to a network create opportunities for new products and services, but also create several classes of problems that need to be solved. Many of these problems or needs in the art pertain to conserving power for a device with transducers and
15 operating a radio, such that a battery in the device can have a long lifetime. A need exists in the art to enable a device to conserve power by minimizing the time that a processor and a radio in a device are in active states and maximizing the time the processor and radio in a device are in sleep states.

Many devices can utilize or require nonvolatile memory in order record data within the
20 device when the device is in a sleep, deep sleep, or even powered off state. The data required can include an operating system for the device. Additional data for a nonvolatile memory could be a logged record of transducer data, in order for the device to perform analysis or send summary information to a server. In addition, the device could continue to store transducer data for transmission at a subsequent time if a wireless network is not available. Consequently, a
25 device may need a nonvolatile memory storage unit such as a standard SD card or micro SD card. Many devices also connect with wireless WAN networks instead of WiFi networks, such as tracking devices that move, equipment that might frequently change location such as medical equipment in an ambulance, and devices monitoring the temperature or environment for food supplies, and other examples exist as well. Thus, many devices also need a wireless WAN radio
30 to connect with a wireless network, such as a network operated by a mobile network operator.

Conventional technology for IoT/M2M devices utilizes a separate storage unit (e.g. microSD card) and a separate radio module. A need exists in the art for a combined "storage radio unit", such that the storage radio unit includes both nonvolatile memory and a radio to connect with a wireless WAN. SD cards with WiFi radios and near field communication (NFC)

radios have been introduced into the market, and conventional technology for SD cards with radios supports local area networks. A need exists in the art for a microSD card to support data transfer through wireless WAN networks such as traditional mobile networks. A need exists in the art for a microSD card to support data transfer through NB-IoT wireless networking technology.

The wide variety of operating systems for devices with transducers creates significant challenges for a manufacturer of radio WAN modules for the devices. A radio module for a wide variety of devices may need to include many different versions of drivers in order to support the different operating systems. Developing and supporting the different drivers can create significant additional overhead or resource requirements for a manufacturer of radio modules. Therefore, a need exists in the art for a device and a storage radio unit to communicate with each other via established and widely deployed standards, such that new software drivers do not need to be created, distributed, and supported in order to enable communication between devices and a storage radio unit. A need exists in the art for a storage radio unit with a radio to communicate with a device using SD card drivers, since the vast majority of operating systems for devices will automatically support SD cards and related drivers.

Connecting a device to a wireless WAN such as supported by a mobile network operator traditionally requires the use of a Subscriber Identity Module (SIM) card. The use of a SIM card with an IoT/M2M device may not be convenient for many reasons, including lack of space, a desire for the device to be hermetically sealed, end users may prefer devices received to be “ready to connect” and not require additional handling of inserting a SIM card. Consequently, a need exist in the art for a storage radio unit that provides both nonvolatile memory and a WAN radio to also support an embedded universal integrated circuit card (eUICC).

The use of an eUICC for a storage radio unit can also create other challenges for users of devices. An eUICC according to GSMA M2M eUICC specifications requires a subscription manager secure routing (SM-SR) function in order to receive profiles, where a profile for the eUICC provides the equivalent of SIM card functionality. A device user or entity deploying the devices may not operate or support an SM-SR, which requires technical expertise and specialized server equipment, in addition to contractual agreements with mobile network operators and authorizations from the GSMA. Consequently, a need exists in the art for a storage radio unit with an eUICC to be provided with integrated support from an SM-SR, such that a device user could simply request an operational profile. A need exists in the art for the storage radio unit to include a bootstrap profile, such that the device automatically has connectivity wherever deployed. A need exists in the art for the SM-SR function included with

the storage radio unit to support changing the bootstrap profile to an operational profile based on requirements of the user for the device with the storage radio unit.

Narrowband wireless WAN technologies such as NB-IoT may not be able to efficiently use traditional IETF security protocols such as Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). A device operating at low power to conserve battery may have a few tens of kilobits per second of bandwidth and hundreds of milliseconds of round-trip delay with a server communicating with the device. A full DTLS handshake may require ~3KB and several seconds using conventional technology with a device connected to an NB-IoT network. A device may have only a hundred bytes or less to transmit in a report. Thus, conducting a DTLS session for each device report may waste considerable energy and bandwidth resources (e.g., > 90% of bandwidth and energy used for overhead). TLS would take longer and more bandwidth, given the additional overhead of TLS compared to DTLS.

A device may transmit reports an exemplary few times a day or less frequently, which can be outside the allowed session resume times for TLS and DTLS, and thus new TLS and DTLS sessions have to be established for each report. Consequently, a need exists in the art for the device or a storage radio unit to secure communication with a server in a manner that does not require standard TLS and DTLS. A need exists in the art for a server to convert the secured communications to standard security protocols such as TLS in order to communicate with an application server for the device, where the application server can support standard security protocols such as TLS.

And other needs exist in the art as well, as the list recited above is not meant to be exhaustive but rather illustrative.

SUMMARY

Methods and systems are provided for (i) operating a radio unit (RU) within a storage radio unit (SRU) and (ii) using a shared nonvolatile memory interface for communications between (a) a device operating the SRU and (b) the RU. The storage radio unit can comprise the form factor of a micro SD card or an integrated circuit soldered to a circuit board. The storage radio unit can also be referred to as a storage memory unit. A system supporting the communication for the device can include a wireless network, a SRU service provider, and a device service provider. The SRU service provider (SRU-SP) can operate a first server "server A", and the device service provider can operate a second server "server B". The wireless network, server A, and server B can be connected using the public Internet. The SRU-SP and the SRU can record a set of compatible cryptographic algorithms and cryptographic parameters, and the SRU-SP can also record at least one elliptic curve cryptography (ECC) private key. An objective of various embodiments of the disclosure is to address the needs in the art described above.

For an exemplary embodiment, a device can include electrical components of a processor, a transducer, random access memory (RAM), a battery, an antenna, a storage radio unit, and device service provider certificate, where the certificate includes a device service provider public key. The device can comprise a computing device to support M2M and IoT applications, in order to collect data regarding the state of a monitored unit and send the data to the second server, server B. The device can operate within range of a wireless wide area network, such as a network for a mobile network operator. The mobile network operator could support wireless narrowband technologies such as standards for "narrowband Internet of Things" (NB-IoT), which could be a subset of the "Long Term Evolution" LTE 4G standards. NB-IoT could also be supported by 5G networks as well.

Upon boot, the device can use an operating system with a driver for the storage radio unit, and libraries for (i) collecting data from a transducer, (ii) reporting data to a server in the form of application data, and (iii) writing the application data to the storage radio unit. The device can power the storage radio unit. The device can collect transducer data and write application data for server B to the storage radio unit. The processor for the device can enter and remain in a sleep state (a) after the device writes the application data to the storage radio unit until (b) SRU signals device 101 that the SRU has a server B response in a shared nonvolatile memory for the device to read. The processor for the device using this configuration for (i) entering and exiting the sleep state and (ii) using shared nonvolatile memory to read the server B response can conserve power. In other words, the processor can sleep for substantially

the entire time between (a) writing application data in a shared nonvolatile memory and (b) server B response data is available for reading by the processor in the device.

A storage radio unit can be manufactured with nonvolatile memory, a radio unit, an external bus, an internal bus, a memory processor, a memory core interface, RAM, an eUICC, a RF clock, EEPROM, a power control unit, and electrical connection to the antenna in the device. The nonvolatile memory can include shared memory for the device and the SRU, general memory for the device, and protected memory for the SRU and radio unit. Upon boot or power up of the SRU, the memory processor can load boot firmware from the EEPROM and then SRU firmware recorded in the protected memory for the SRU. After the boot process, the memory processor can subsequently communicate with (i) the device using the external bus and (ii) internal components in the SRU using the internal bus. The protected memory for the SRU can record an elliptic curve cryptography (ECC) SRU private key and an SRU-SP public key corresponding to the SRU-SP private key.

The radio unit in the SRU can include a processor, a RF front-end connected to the device antenna, a RF transceiver unit, and a baseband unit. For some embodiments, the RF-front-end can be external to the SRU and connected to the SRU via a data bus. The timing of the radio unit can be driven by the RF clock. The radio unit can be connected to the memory processor and the nonvolatile memory in the SRU using an internal bus. The radio unit can conduct a boot process and load RU firmware from the protected memory of the SRU. The power control unit in the SRU can control the RU and instruct the RU to enter a sleep state after boot.

The SRU firmware loaded by the memory processor can include libraries for the SRU to access and control nonvolatile memory in the SRU via the memory core interface and the internal bus. The SRU firmware can also include a configuration process. The configuration process can read and detect the type of filesystem written to the general memory for the device. The configuration process can use filesystem drivers in the SRU firmware to configure the shared nonvolatile memory with the detected filesystem. The configuration process can write a device write and SRU read directory in the shared nonvolatile memory. The configuration process can write a SRU write and device read directory in the shared nonvolatile memory. The device write directory and the SRU write directory can comprise a shared filesystem interface between the device and the SRU. The configuration process can then load an initial set of files in the shared nonvolatile memory. The operating system for the device can mount the filesystem for the shared nonvolatile memory and the device general memory. The device can

access the shared nonvolatile memory and the device general memory using a standard driver for the operating system of the device, such as an SD card driver.

The device can write the application data to the device write directory in the shared filesystem interface operating in the nonvolatile memory of the SRU. The memory processor
5 can process a trigger library from the SRU firmware to detect that application data has been written to the shared filesystem interface. The trigger library can instruct the power control unit to change the state of the radio unit from a sleep state to an active state. Upon entering the active state, the radio unit can attach to the wireless network using credentials recorded in a profile for the eUICC. The radio unit (RU) can then read the application data from the device
10 write directory.

The RU can derive an ECC ephemeral PKI key pair comprising an ephemeral private key and an ephemeral public key. The RU can conduct an ECDH key exchange using as input the ephemeral private key, the SRU private key, and the SRU-SP public key. The output of the ECDH key exchange can comprise a secret shared key that also can be mutually derived by
15 server A of the SRU-SP. The secret shared key can be input into a key derivation function in order to derive a symmetric ciphering key. The RU can compress the application data using a compression library included in SRU firmware. The RU can encrypt the compressed application data using the derived symmetric ciphering key and a symmetric ciphering algorithm. The RU can transmit a message to server A through the narrowband or wideband wireless network
20 comprising (i) the encrypted, compressed application data along with (ii) the derived ephemeral public key and (iii) an identity of the SRU.

Server A of the SRU-SP can record an SRU public key corresponding to the SRU private key recorded by the SRU. Server A can receive the message from the RU via the wireless network and the Internet. Server A can read the SRU identity in the message and lookup the
25 SRU public key as well as cryptographic parameters associated with the SRU and SRU public key. For some embodiments, the SRU identity can comprise a secure hash value over a static public key for the SRU. The SRU can read the ephemeral public key received in the message. The SRU-SP can conduct an ECDH key exchange using the received ephemeral public key, the SRU public key, and the SRU-SP private key in order to derive the secret shared key. The SRU-
30 SP can input the secret shared key into a key derivation function in order to derive the symmetric ciphering key. The SRU-SP can decrypt the encrypted, compressed application data using the derived symmetric ciphering key and a symmetric ciphering algorithm. The SRU-SP can decompress the application data in order to read the plaintext application data. The SRU-SP

can establish a TLS session with server B. The SRU-SP can forward the application data in the TLS session to server B.

Server B can receive the application data for the device. The application data can include a device identity. Server B can process the data and generate a server B response along with a digital signature for the server B response data using a SRU-SP private key. Server B can send Server A for the SRU-SP the server B response and the digital signature for the server B response data through the TLS session. Server A can map or track which sessions with server B are associated with which SRU via the SRU identity received in the message from the RU. The SRU-SP can receive the server B response and compress the data. The SRU-SP can encrypt the compressed data using the derived symmetric ciphering key. The SRU-SP can forward (a) the encrypted, compressed server B response data and (b) the digital signature for the server B response data to the RU via the Internet and the narrowband or wideband wireless network.

The RU in the SRU can receive (a) the encrypted, compressed server B response data and (b) the digital signature. The RU can decrypt the encrypted, compressed server B response data using the derived symmetric ciphering key and the symmetric ciphering algorithm. The RU can decompress the compressed server B response data. The RU can write (a) the server B response data and (b) the digital signature to the SRU write and device read directory in the shared nonvolatile memory. The power control unit can (i) return the RU to a sleep state and (ii) signal the device for the device to wake the processor in the device or change the processor in the device from the sleep state to the active state. The device can read the server B response data and the digital signature for the server B response data from the shared nonvolatile memory. The server B response data could include an actuator setting or other command for the device, or simply an acknowledgement that the application data was received. The device can verify the digital signature using the certificate recorded for the device service provider (which operates server B). After reading and processing the server B response data, the device can optionally delete the data from the shared nonvolatile memory.

For another exemplary embodiment, a device can include the radio unit and nonvolatile memory, without requiring the use of a storage radio unit. The device or a configuration unit can conduct a configuration process on the nonvolatile memory to configure a shared nonvolatile memory with a device write memory and a RU write memory. The device can write application data to the device write memory. The device can operate a power control unit and change the state of the RU from a sleep state to an active state after the device writes the application data. A processor in the device can then enter a sleep state until the radio unit writes server B response data to the RU write memory. The RU can wake from a sleep state and

connect with a wireless network using an eUICC. The RU can read the application data from device write memory and conduct the steps for a RU listed above for the RU to transmit and send the application data.

Server A and server B can conduct the steps listed above for receiving and processing the application data. Server A and server B can conduct the steps for processing and communicating the server B response data and a digital signature from server B for the server B response data. The RU in the device can (i) receive the server B response data and digital signature via the wireless network and (ii) write the server B response data and the digital signature to the RU write memory. The RU in the device can signal the processor in the device to exit from a sleep state in order for the processor in the device to read the server B response data and the digital signature. The device can then verify the digital signature over the server B response data using a recorded certificate associated with server B. The device can then process and operate with the server B response data.

For another exemplary embodiment, a device can include a system on a chip, where the system on the chip includes a processor, nonvolatile memory, and a radio unit. The components in the system on the chip can be connected via an internal system bus. For embodiments with the “system on a chip” configuration, some components for an RU used in a storage radio unit configuration may be operated outside the system on a chip, such as an RF front end operated outside the system on a chip. An RF clock within the system on the chip can drive the internal timing of both the processor and the internal radio units. The nonvolatile memory can comprise a persistent memory or a nonvolatile RAM (NVRAM) memory. The system on a chip can operate with the steps for a device according to the steps for a device and a RU listed above in the previous two paragraphs, where (i) the processor writes application data for the RU to the nonvolatile memory in the system on a chip and (ii) the RU writes server B response data and a digital signature for the processor to the nonvolatile memory in the system on a chip.

The processor can signal the RU to wake after writing the application data and also enter a sleep state. The RU can signal the processor to wake after writing the server B response data and the digital signature, and also return to a sleep state. The processor can read the server B response data and the digital signature, verify the digital signature using a certificate associated with server B, and process and operate with the server B response data. In some exemplary embodiments, the server B digital signature can be omitted, and the processor can conduct encryption of the application data before writing the application data for the RU to the shared nonvolatile memory.

These as well as other aspects and advantages will become apparent to those of ordinary skill in the art by reading the following detailed description, with reference where appropriate to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

5 Various exemplary embodiments are described herein with reference to the following drawings, wherein like numerals denote like entities.

Figure 1a is a graphical illustration of an exemplary system, where a storage radio unit operates with a device, and the storage radio unit communicates with a service provider using a radio and a wireless network, in accordance with exemplary embodiments;

10 Figure 1b is a graphical illustration of hardware, firmware, and software components for a device, including a storage radio unit, in accordance with exemplary embodiments;

Figure 1c is a graphical illustration of hardware, firmware, and software components for a device and a storage radio unit, in accordance with exemplary embodiments;

15 Figure 2a is a graphical illustration of components within a storage radio unit, in accordance with exemplary embodiments;

Figure 2b is a graphical illustration of an exemplary system, where a device and a radio unit communicate through a shared nonvolatile memory in a storage radio unit, in accordance with exemplary embodiments;

20 Figure 2c is a graphical illustration of a storage radio unit utilizing two clock inputs, in accordance with exemplary embodiments;

Figure 2d is a graphical illustration for using an interface between a device and a storage radio unit, in accordance with exemplary embodiments;

Figure 2e is a graphical illustration of storage radio unit firmware 109u, in accordance with exemplary embodiments;

25 Figure 2f is a flow chart illustrating exemplary steps for configuring a storage radio unit to communicate with a device using a shared nonvolatile memory interface, in accordance with exemplary embodiments;

Figure 3a is a graphical illustration of a shared filesystem interface for a radio storage unit and a device, using a device write directory, in accordance with exemplary embodiments;

30 Figure 3b is a graphical illustration of a shared filesystem interface for a radio storage unit and a device, using a radio unit write directory, in accordance with exemplary embodiments;

Figure 3c is a flow chart illustrating exemplary steps for a device and a storage radio unit to utilize a shared filesystem interface, in accordance with exemplary embodiments;

Figure 4a is a simplified message flow diagram illustrating an exemplary system with exemplary data sent and received by a device, a storage radio unit, a radio unit, a server A, and a server B, in accordance with exemplary embodiments;

5 Figure 4b is a flow chart illustrating exemplary steps for conducting a key exchange using PKI keys in order to derive a shared secret key and a symmetric ciphering key, in accordance with exemplary embodiments;

Figure 5a is a graphical illustration of hardware, firmware, and software components for a device and a storage radio unit, including a shared memory interface operating in RAM for the storage radio unit, in accordance with exemplary embodiments;

10 Figure 5b is a flow chart illustrating exemplary steps for configuring a storage radio unit to communicate with a device using a shared nonvolatile memory interface in RAM, in accordance with exemplary embodiments;

15 Figure 6a is a flow chart illustrating exemplary steps for storage radio unit to communicate with a device using a shared nonvolatile memory interface, in accordance with exemplary embodiments;

Figure 6b is a flow chart illustrating exemplary steps for a storage radio unit to operate a radio unit and a shared nonvolatile memory interface, in accordance with exemplary embodiments;

20 Figure 7 is a graphical illustration of hardware, firmware, and software components for a device, in accordance with exemplary embodiments; and,

Figure 8 is a graphical illustration of hardware, firmware, and software components for a device, in accordance with exemplary embodiments.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS OF THE INVENTION**Figure 1a**

Figure 1a is a graphical illustration of an exemplary system, where a storage radio unit operates with a device, and the storage radio unit communicates with a service provider using a radio and a wireless network, in accordance with exemplary embodiments. The system 100 includes a device 101, a wireless network 102, a storage radio unit service provider (SRU-SP) 106, and a device service provider 110. Device 101 can include a storage radio unit (SRU) 109 and an antenna 101t for connecting to the wireless network 102. Device 101 can be a wireless computing device to support applications for the “Internet of Things” or “Machine-to-Machine” communications including tracking devices, security systems, cameras, or health monitoring devices. Device 101 can also be a computing device, a mobile phone, a module, a drone, a laptop computer, a desktop computer, a server, a tablet, a “point of sale” cash register or payment terminal, and many other possibilities exist as well. Device 101 can be associated with or operate in conjunction with a monitored unit 105. Exemplary electrical and physical components operating within device 101 are also depicted and described in connection with Figure 1b and Figure 1c below. Although a single device 101 is depicted in Figure 1a, a wireless network 102 could include a plurality of devices 101.

Device 101 can be associated with a monitored unit 105. Examples of a monitored unit 105 can include an ATM or vending machine, a truck or automobile, a refrigerated or standard (“dry”) shipping container, or industrial equipment such as, but not limited to, an oil field pump, a transformer connected to an electrical grid or an escalator or elevator in a building. Additional examples of a monitored unit 105 include can also include a pallet for shipping or receiving goods, a refrigerator with food, a health monitoring device attached to a person such as, but not limited to, a heart monitor, and a gate or door for opening and closing. Device 101 can utilize a sensor (depicted in Figure 1b) to measure and collect data regarding a parameter of monitored unit 105 such as, but not limited to, temperature, humidity, physical location potentially including geographical coordinates from a Global Positioning System (GPS) or other satellite system receiver, surrounding light levels, surrounding RF signals, vibration and/or shock, voltage, current, audio or sound, and/or similar measurements. Other possibilities exist as well for device 101 to associate with a monitored unit 105 without departing from the scope of the present disclosure. In some exemplary embodiments, a monitored unit 105 could be omitted,

and data from a device 101 directly could be communicated through wireless network 102 to a server such as server A 108 and server B 110a.

Monitored unit 105 could also be controlled by device 101 via a transducer that is an actuator (depicted in Figure 1b), where device 101 can change the actuator in order to change a state or parameter for monitored unit 105. For example, if monitored unit 105 is a door, then a transducer for device 101 could include a relay to activate a lock for the door. If monitored unit 105 is a lighting system in a building, then a transducer could comprise a switch to turn the lighting level up or down. Other examples exist for monitored unit 105 as well, and the above are intended to be illustrative instead of limiting. Device 101 could forward sensor data to device service provider 110 as application data 112a. Device service provider 110 can both (i) receive application data 112a from sensors in device 101 for monitored unit 105 and (ii) transmit server B response data 112b for actuators in device 101 in order to control monitored unit 105. Thus, in operation and by using the systems and methods described in subsequent figures below, device service provider 110 can remotely monitor and control monitored unit 105 using device 101, where device 101 obtains connectivity to wireless network 102 via SRU 109, or a radio unit operating in device 101.

Wireless network 102 can comprise a wide area wireless network operated by a mobile network operator and support wireless wide area network (WAN) technologies including 3rd Generation Partnership Project's (3GPP) 3rd Generation (3G) Universal Mobile Telecommunications System (UMTS) and 4th Generation (4G) Long-term Evolution (LTE), LTE Advanced, Narrow-band Internet of Things (NB-IoT), and 5G standards as well, including subsequent standards. Wireless network 102 could also support other wireless standards such as low power wide area networking such as Sigfox and LoRa. Wireless network 102 can operate a base station 103, which could comprise an "eNodeB" (for 4G networks), a "gNodeB" (for 5G networks), a picocell, a macrocell, or other wireless WAN radio access network technologies. For some exemplary embodiments, base station 103 could comprise an access point for an 802.11 network, such as, but not limited to, and 802.11 ac, 802.11 ax, or 802.11 ah network.

Wireless network 102 can require device 101 to record and operate with (i) a set of network access credentials such as an international mobile subscriber identity (IMSI) and key K for 4G networks, and/or (ii) a subscription permanent identity (SUPI) and master key for 5G networks, and other possibilities exist as well. Network access credentials for wireless network 102 can be recorded in an embedded universal integrated circuit card (eUICC) 113 in SRU 109 in device 102. Although a single wireless network 102 is depicted in Figure 1a, a system 100 could include a plurality of wireless networks 102. In addition, wireless network 102 could

support WiFi wireless networking, including, but not limited to, 802.11 based standards, such as “ac”, “ah”, and future or related versions of WiFi standards as well including 802.11 ax.

Device 101 can communicate with SRU-SP 106 via the wireless network 102. SRU-SP 106 can comprise an entity, business, or collection of computing devices associated with SRU 109 and providing data communications services to device 101 via SRU 109. SRU-SP 106 can include a Subscription Manager – Secure Routing (SM-SR) 107 function according to GSM Association standard SGP.02 version 3.2 titled “Remote Provisioning Architecture for Embedded UICC Technical Specification”, which is herein incorporated by reference in its entirety. SM-SR 107 can record information pertaining to eUICC 113 in SRU 109, including an eUICC information set (EIS). SM-SR 107 could also communicate with a plurality of different Subscription Manager – Data Preparation (SM-DP) functions (not shown), where an SM-DP could be associated with wireless network 102. The SM-SR 107 can establish a secure channel with the eUICC 113 and transfer profiles to the eUICC through the secure channel, according to the specification in GSMA SGP.02. The SM-SR 107 can receive profiles for eUICC 113 from a SM-DP, based on requests or contractual agreements between the SM-SR 107 and the SM-DP.

SRU-SP 106 can also include at least one server A 108 for communicating application data 112a and server B response 112b data with SRU 109 via the wireless network 102. Server A 108 can comprise a server in order to communicate with a plurality of SRU 109 in different devices 101. Different devices 101 could operate with different wireless networks 102. Server A 108 and server B 110a for both SRU-SP 106 and device service provider 110 can be either different physical computers such as rack-mounted servers, or different logical or virtual servers or instances operating in a “cloud” configuration. Internal electrical and physical components for server A 108 and server B 110a can be equivalent to those depicted and described for a device 101 in Figure 1b and Figure 1c below, but with high capacities associated with servers. Server A 108 and server B 110a for both SRU-SP 106 and device service provider 110 can each be co-located within the same data center or geographically dispersed.

Server A 108 and server B 110a for both SRU-SP 106 and device service provider 110 can operate with an operating system for a server such as Linux or Windows Server, and other possibilities exist as well. Although a single server A 108 and server B 110a is depicted in Figure 1a, server A 108 and server B 110a could each separately comprise a plurality of different servers or computing devices with different functions to operate collectively as a single server, such as a firewall, a database, a router, and an application such as a web server or an email server. In addition, although server A 108 and server B 110a are depicted as operating

with separate entities, the function of server A 108 and server B 110a could be combined into a single server operating with one of the two entities SRU-SP 106 or device service provider 110.

Server A 108 can include a SRU public key 109w for each SRU 109, a device service provider certificate 110c for device service provider 110, and an SRU-SP private key 106a. SRU public key 109w can be selected based on an identity for SRU 109, and SRU public key 109w can also be recorded in a certificate. Certificates in the present disclosure can be formatted according to the X.509v3 and related standards, and record information about the public key, unit holding the corresponding private key, and cryptographic parameters associated with the public and private keys. In exemplary embodiments, public and private keys can be used with cryptographic algorithms supporting key exchange and digital signatures. Public and private keys can support either elliptic curve cryptography (ECC) or RSA-based cryptographic algorithms, and other cryptographic algorithms with public and private keys can be supported as well, including cryptographic algorithms and PKI keys supporting post-quantum cryptography, without departing from the scope of the present disclosure.

Device service provider 110 can comprise an entity, business, or collection of servers providing service to device 101. Device service provider 110 can include device service provider private key 110b, server B 110a, and configuration unit 110d. Device service provider private key 110b can comprise a private key corresponding to device service provider public key 110c (or certificate) recorded by SRU-SP 106 in server A 108. Server B 110a can comprise a server or computing device to receive data from device 101 over an IP network 111 and can also include data regarding monitored unit 105. Although not depicted in Figure 1a, server B 110a can include a database recording application data 112a for a plurality of different devices 101.

Server B 110a can also function as a monitoring and controlling service for device 101, such as logging data and also sending commands or instructions to device 101 in server B data 112b. Server B 110a could support a wide variety of applications, such as tracking location of device 101, monitoring physical security status reported by device 101, monitoring health data for a user associated with device 101, monitoring the status or state of monitored unit 105, and other possibilities exist as well without departing from the scope of the present disclosure. In exemplary embodiments, server B 110a can be referred to as an application server. Server B 110a could support web interfaces to users for users or owners of device 101 to monitor the status of device 101. Although Figure 1a depicted server B data 112b for device 101 as being transmitted through SRU-SP 106, in some embodiments application data 112a and server B data 112b could be communicated between device 101 and device provider 110 without passing through SRU-SP 106.

In some embodiments, device service provider 110 can operate both server A 108 and server B 110a. For these embodiments, SRU-SP 106 could continue to operate SM-SR 107 but not operate server A 108. For these embodiments, device service provider 110 obtains the benefits for the use of a server A 108, where communication and encryption of data between server A 108 and RU 109b can comprise a first protocol optimized for an NB-IoT 104 network or other narrowband wireless network technology. Communication between server A 108 and server B 110a can utilize standard IETF security protocols such as TLS between the two servers. In other words, TLS or DTLS may not be optimal or preferred for the bandwidth and power/energy requirements of device 101 with RU 109b operating on a battery and with narrowband wireless technologies. Consequently, a first server comprising server A 108 can manage the communication with RU 109b over the narrowband wireless technology using a first protocol for communication and security, and server A 108 can manage the communication and security with server B 110a using a second protocol for communication and security, such as TLS. Additional details regarding communication between RU 109b, server A 108, and server B 110b are provided below in Figure 4b and related figures.

Server B 110a can include an Internet Protocol (IP) address 110e, which could comprise an address according to IPv4 or IPv6 standards. Other servers and computing devices depicted in Figure 1a could include an IP address as well and connect to an IP network 111. Data between the nodes and entities depicted in Figure 1a can be communicated via the IP network 111 using IP addresses such as IP address 110e. IP network 111 can comprise either the globally routable public Internet or also a private network which could be an intranet. IP network 111 could comprise a combination of intranets linked to the globally routable public Internet.

Configuration unit 110d for device service provider 110 can comprise a device for (i) configuring device nonvolatile memory 109a in SRU 109 and (ii) recording firmware, software, and data for device 101 in the configured memory 109a. By inserting SRU 109 in configuration unit 110d before SRU 109 is inserted into device 101, memory 109a can record data for device 101 before device 101 begins operation. Although not depicted in Figure 1a (but depicted and described in connection with Figure 1c and Figure 2a below), (x) memory 109g for device 101 in SRU 109 can reside within nonvolatile memory 109a in SRU 109, and (y) a configuration unit 101d could configure memory 109g in memory 101a. The use of a configuration unit 110d can be omitted in some embodiments, and a device 101, user or SRU 109 manufacturer can configure memory 109g for those embodiments. Although a configuration unit 110d is depicted

in Figure 1a as associated with a device service provider 110, for some exemplary embodiments a SRU-SP 106 could operate a configuration unit 110d.

Storage radio unit 109 can comprise a computing device that can be inserted into device 101. In exemplary embodiments, storage radio unit (SRU) 109 can comprise a Secure Digital (SD) card supporting physical standards maintained by the SD Card Association (SDA). SRU 109 can have a physical form compatible with (i) standard SD cards and dimensions of 32 mm x 24 mm x 2.1 mm, or (ii) micro SD card and dimensions of 15 mm x 11 mm x 1.0 mm. Other possibilities exist as well for the physical form of SRU 109, including the “miniSD” size. SRU 109 could also comprise an integrated circuit that could be soldered onto a circuit board for device 101. SRU 109 could also comprise a wafer level chip scale package (WLCSP), which could be included in a separate integrated circuit, such as a “system on a chip” for device 101 (as depicted in Figure 8 below). Note that in some exemplary embodiments, the components depicted for storage radio unit 109 could be operated by other electrical components within device 101, such as depicted in Figure 7 and Figure 8 below, and in this case an SRU 109 could be omitted while obtaining the benefits and functionality of the present disclosure, including the use of a shared nonvolatile memory interface for radio unit 109b and device 101 as described below in Figure 2b, Figure 2f, Figure 3a, Figure 3b, etc.

Storage radio unit 109 can include nonvolatile memory 109a, radio unit 109b, SRU private key 109p, SRU-SP public key 106b, and eUICC 113. In exemplary embodiments, each of nonvolatile memory 109a, radio unit 109b, SRU private key 109p, and eUICC 113 comprise either (i) wafer level chip scale packages (WLCPS) or (ii) separate components etched and processed within a single silicon chip or package, in order to support a micro SD card form factor. SRU private key 109p can comprise the private key corresponding to SRU public key 109w recorded with SRU-SP 106 and described in this Figure 1a above. SRU-SP public key 106b can be the public key corresponding to the SRU-SP private key 106a recorded and operated by server A 108 with SRU-SP 106.

Non-volatile memory 109a can comprise a physical memory of NAND or NOR flash memory, such that data recorded in non-volatile memory 109a continues to be recorded when electrical power is removed from storage radio unit 109. The data within non-volatile memory 109a can subsequently be read and/or re-written when power is restored to storage radio unit 109. Exemplary values for the amount of memory available non-volatile memory 109a for SRU 109 operating as an SD card physical form factor can range from 512 megabytes to 128 gigabytes, and other possibilities exist as well. For some embodiments, nonvolatile memory

109a can comprise a persistent memory or nonvolatile RAM memory (NVRAM), such as that depicted in Figure 8 below.

Non-volatile memory 109a can include occasional bit errors due to the nature of the physical memory, such as small memory cell sizes, but error correcting codes operating in a memory processor in SRU 109 (depicted as processor 109d in Figure 1c below) can normally correct the errors, such as limiting errors in a file read by device 101 from non-volatile memory 109a to less than one part per billion during the normal, operating lifetime of storage radio unit 109. As contemplated herein, error correcting codes can comprise either (i) convolution codes operating on a bit by bit basis on memory or data from memory, such as a Viterbi decoder, or (ii) block codes, such as a Hamming codes or Reed-Solomon codes. Other possibilities for the physical structure of non-volatile memory 109a and error correcting codes exist as well without departing from the scope of the present disclosure, and generally (i) non-volatile memory 109a includes addresses and blocks, such that binary data can be recorded and subsequently read in a nonvolatile state, and (ii) error correcting codes attempt to identify and correct the presence of bit errors in either physical memory and/or data read from the physical memory.

Radio unit 109b operating in SRU 109 can comprise a radio with standard radio components such as RF filters, RF amplifiers, a clock, a controller or processor, and phased loop logic (PLL), and may be connected with an antenna such as antenna 101t for device 101. Additional details for radio unit 109b are depicted and described in connection with Figure 2a below. In exemplary embodiments, radio unit 109b supports the wireless WAN standards for wireless network 102, which could comprise a network supporting NB-IoT 104 radio frequency signaling and standards. NB-IoT was initially standardized with 3GPP Release 13, and large mobile networks in the United States such as T-Mobile and AT&T are deploying NB-IoT networks nationwide throughout 2018 and 2019. Benefits of NB-IoT 104 communications for radio unit 109b include low power and high sensitivity, such as using sub-channels or carriers totaling ~180 kHz and bit rates on the order of ~100 kbps, or lower for “deep penetration” in buildings or remote locations. Other narrowband wireless technologies could be used with RU 109b and wireless network 102 as well.

In exemplary embodiments, radio unit 109b supports subsequent and related standards for NB-IoT 104, including updates through 3GPP Release 16, which is the latest in formal draft phase as of 2018. Radio unit 109b could also support other relatively low electrical power and low processing power standards or protocols, including Sigfox and Lora. In general, the radio protocols supported by radio unit 109b are characterized by (i) low transmit power such as a maximum transmit power of 30 dBm or less, and (ii) relatively narrow frequency channels such

as less than 2 MHz, and preferably approximately equal to less than 200 kHz. For NB-IoT applications, narrow sub-channels or carriers such as 15 or 3 kHz could be utilized. For some embodiments, wireless network 102 and RU 109b could support LTE CAT M1, which comprises using approximately 1 MHz of RF spectrum. RU 109b could operate in either half duplex mode or a full duplex mode. The RF bands for radio unit 109b could use either frequency division duplex (FDD) or time division duplex (TDD).

eUICC 113 operating with SRU 109 can operate according the GSMA standards in SGP.02 for an eUICC, including subsequent and related standards. eUICC 113 can be manufactured by an eUICC Manufacturer (EUM) certified by the GSMA. eUICC 113 can include an Issuer Security Domain – Root (ISD-R) and an eUICC Controlling Authority Security Domain (ECASD), along with an internal processor and memory and a bus. eUICC 113 could also comprise a secondary platform bundle operating within a “Smart Secure Platform” within SRU 109. Separate profiles can be recorded in Issuer Security Domains – Profile (ISD-P), which can be created based on commands from the SM-SR 107 and SM-DP associated with wireless network 102. A profile can contain the data equivalent to that recorded in a traditional SIM card comprising network access credentials (e.g. IMSI and key K) and wireless network 102 parameters (e.g. frequencies or channels to use, network codes, timers to implement, etc.)

In exemplary embodiments, eUICC 113 includes a bootstrap profile 113c and an operational profile 113d. Bootstrap profile 113c can be written to eUICC 113 by the SM-SR 107 function, or specified by SM-SR 107 and written by the EUM. For some embodiments, eUICC 113 could use GSMA standards SGP.22 for consumer devices (where device 101 can include a user interface 101j as depicted in Figure 1b below). Use of SGP.02 may be preferred if device 101 does not operate a user interface. In exemplary embodiments, eUICC 113 can support both SGP.02 and SGP.22.

Bootstrap profile 113c provides initial or fallback connectivity on a global basis, and could be associated with a globally operating wireless network 102 such as AT&T or Vodafone. Operational profile 113d could be downloaded or updated using the steps specified for a profile download procedure with an eUICC from GSMA standards. Operational profile 113d could be used by SRU 109 and device 101 based on updated contractual agreements with a wireless network 102 or for use in specific geographies. Although a single eUICC 113 is depicted for SRU 109 in Figure 1a, in some exemplary embodiments an SRU 109 could include multiple different eUICC 113 and select the eUICC 113 based on preferences in SRU 109 firmware (such as SRU firmware 109u in Figure 1c, 2a, 2e, etc. below). Benefits of SRU 109 containing

multiple eUICC 113 units is that SRU 109 can select a profile to use from each of the multiple different eUICC 113 based on the operational profile 113d in each of the different eUICC 113.

According to the current GSMA specifications for eUICCs in embedded devices (such as with SGP.02), an operational profile 113d cannot be changed without a command from or interaction with an SM-DP (except for fallback or bootstrapping cases). By using multiple different eUICC 113 in SRU 109, the SRU 109 can select a different eUICC 113 with different operational profiles 113d in each eUICC 113 in order to have access to multiple profiles without involving an SM-DP. The selection of an eUICC 113 by SRU 109 is described with step 406a in Figure 4 below. An eUICC 113 can also be referred herein and within related literature as an “integrated universal integrated circuit card” (iUICC), as depicted with Figure 8 below. In exemplary embodiments, the same SM-SR 107 could be used with each of the different eUICC 113 operating in device 101. Or, in some embodiments a single eUICC can be used with SRU 109 and device 101.

Using the components depicted for a system 100, a device 101 with an SRU 109 and a radio unit 109b can connect with a first wireless WAN network 102, such as using NB-IoT 104 standards. Wireless WAN network 102 can provide access to IP network 111 and SRU-SP 106 and/or device service provider 110. The SM-SR 107 function of SRU-SP 106 can support sending an operational profile 113d to the eUICC 113 operating in SRU 109. SRU 109 can use the operational profile 113d to connect with a second wireless network 102, based on the network access credentials and network parameters in the downloaded operational profile 113d. Device 101 can then send application data 112b to SRU-SP 106 via the second wireless network 102 and IP network 111. SRU-SP 106 can establish a secure session with device service provider 110 such as TLS session 114. SRU-SP 106 can then forward the application data 112a to device service provider 110 via a TLS session 114.

The device service provider 110 can receive and process the application data 112a from the TLS session 114 using server B 110a. The device service provider 110 can send server B response 112b data to SRU-SP 106 via the TLS session 114. The SRU-SP 106 can then forward server data 112b for device 101 via the second wireless network 102 and the SRU 109 operating within device 101. Server B response 112b data could be a response or confirmation or command for device 101 based on the application data 112a received by device service provider 110. SRU 109 can write the server B response 112b to an interface operating in nonvolatile memory 109a, which can be subsequently read by device 101. In this manner, device 101 can use an SRU 109 to communicate with device service provider 110.

Note that the use of SRU-SP 106 can support standard IETF protocols such as TLS version 3 (e.g. draft-ietf-tls-tls13-28) through IP network 111 with device service provider 110. However, narrowband wireless network technologies such as NB-IoT may not be suited for TLS due to the low bandwidth and potential higher latency (for “deep” building penetration or remote locations). For example, device 101 may prefer not to establish a TLS or DTLS session with server B 110a each time device 101 sends small reports or data such as an exemplary ~50 or ~100 bytes. Transmitting small sets of application data 112a to server B 110a (via wireless network 102 with NB technology and server A 108) may be better supported including a longer battery life for device 101 through a system 100. Also note that application data 112a can be compressed and encrypted by SRU 109 before transmission to SRU-SP 106 and server data 112b can be decrypted and decompressed by SRU 109 after receipt from SRU-SP. In this manner, a system 100 can address needs in the art described in the background above. Additional details regarding the operation of a system 100 and components will be provided in figures below.

Figure 1b

Figure 1b is a graphical illustration of hardware, firmware, and software components for a device, including a storage radio unit, in accordance with exemplary embodiments. Figure 1b is illustrated to include many electrical components that can be used by device 101. Device 101 may also operate in a wireless configuration in order to connect with a wireless network 102. Device 101 may consist of multiple components in order to collect sensor data or control an actuator associated with a monitored unit 105. The physical interface 101p can include associated hardware to provide connections to components such as user interface 101j, a power supply, external transducers, a display or light-emitting diodes, etc. Device drivers 101g can communicate with the physical interfaces 101p, providing hardware access to higher-level functions on device 101. Device drivers 101g may also be embedded into hardware or combined with the physical interfaces.

Device drivers 101g can include a SD card driver 101k, which can be utilized by a device 101 and operating system 101h in order to read and write data to storage radio unit 109, including communicating with a radio unit 109b within storage radio unit 109. SD card driver 101k can operate (i) an SD bus or (ii) a serial peripheral interface (SPI) for physical interface 101a with SRU 109. Other device drivers and data busses than supporting an SD card can be utilized by a device 101 as well in order to communicate with an SRU 109. Additional details

regarding the communication between device 101 and radio unit 109b within storage radio unit 109 are described in additional figures below, including Figure 2b and Figure 2d. Device 101 may preferably include an operating system 101h to manage device drivers 101g and hardware resources such as physical interface 101p and RAM 101e within device 101. The operating systems described herein can also manage other resources such as processor 101b and may support multiple software programs or software libraries operating on device 101.

The operating system 101h can include Internet protocol stacks such as a User Datagram Protocol (UDP) stack, Transmission Control Protocol (TCP) stack, a domain name system (DNS) stack, etc., and the operating system 101h may include timers and schedulers for managing the access of software to hardware resources, including storage radio unit 109. The operating system shown of 101h can be appropriate for a low-power device with limited memory and CPU resources (compared to a server 106 or 110a). Example operating systems 101h for a device 101 includes Linux, Android® from Google®, iOS from Apple®, Windows® Mobile, or Open AT® from Sierra Wireless®. Additional example operating systems 101h for device 101 include eCos, uC/OS, LiteOs, Contiki, OpenWRT, Raspbian, and other possibilities exist as well without departing from the scope of the present disclosure.

A device program 101i may be an application programmed in a language such as C, C++, Java, and/or Python, and could provide functionality to support M2M applications such as monitoring of sensors and activation or control of actuators. Device program 101i could also be a software routine, subroutine, linked library, or software device, according to one preferred embodiment. As contemplated herein, a device program 101i may be an application operating within a smartphone, such as an iPhone® or Android®-based smartphone, and in this case device 101 could comprise the smartphone. The application functioning as a device program 101i could be downloaded from an “app store” associated with the smartphone or operating system for the device 101. Note that the use of a smartphone for device 101 is not required for some exemplary embodiments.

Device program 101i can include data reporting libraries 101x, which can provide the functionality or CPU 101b instructions for (i) collecting sensor data, and (ii) sending messages such as application data 112a to SRU 109 via a structure of device write directory 191x'. In other words, a structure of device write directory 191x' is different than a device write directory 191x (from Figure 3a below), where device write directory 191x could reside in a shared nonvolatile memory 101h, while the structure of device write directory 191x' in a data reporting libraries 101x can comprise instructions, structure, and format such that data reporting libraries 101x can write to a device write directory 191x in memory 109h.

Similarly, data reporting libraries 101x can store a structure of RU write directory 192x', where a structure is stored in 101x and the actual RU write directory 192x can reside in shared nonvolatile memory 109h. Data reporting libraries 101x can provide CPU 101b instructions for (i) reading server B response 112b data from shared nonvolatile memory 109h. Device program 101i can also include the logic for connecting with a server A 110a (via SRU 109), sending a device identity 101a to the server A 110a, receiving a challenge or nonce from server A 110a, and sending a digital signature for the challenge or nonce to the server, where the digital signature can be calculated using a certificate 110c recorded in device 101.

Note that device 101 may also optionally include user interface 101j which may include one or more devices for receiving inputs and/or one or more devices for conveying outputs. User interfaces are known in the art and generally are simple for devices such as a few LED lights or LCD or OLED display, and thus user interfaces are not described in detail here. User interface 101j could comprise a touch screen if device 101 operates as a smartphone or mobile phone. As illustrated in Figure 1b, device 101 can optionally omit a user interface 101j, since no user input may be required for many M2M applications, although a user interface 101j could be included with device 101.

Device 101 may be a computing device that includes computer components for the purposes of collecting data from a sensor 101f or triggering an action by an actuator 101y. Device 101 may include a central processing unit (CPU) 101b, a random access memory (RAM) 101e, and a system bus 101d that couples various system components including the random access memory 101e to the processing unit 101b. The system bus 101d may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures including a data bus. Device 101 can operate a plurality of different busses 101d such as a memory bus, an I2C bus, a USB bus, a SD bus, and a SPI bus, a serial bus with a universal asynchronous receiver-transmitter (UART), a Peripheral Component Interconnect (PCI) bus, or a Peripheral Component Interconnect Express (PCIe) bus. Other possibilities exist as well for a device 101 to use a bus 101d to connect electrical components without departing from the scope of the present disclosure.

Device 101 may include a read-only memory (ROM) 101c which can contain a boot loader program. Although ROM 101c is illustrated as "read-only memory", ROM 101c could comprise long-term memory storage chipsets or physical units that are designed primarily for writing once and reading many times, such as Electrically Erasable Programmable Read-Only Memory (EEPROM). As contemplated within the present disclosure, a read-only address could comprise an address within a read only memory such as a ROM 101c memory address or

another hardware address for read-only operations accessible via bus 101d. Changing data recorded in a ROM 101c can require a technician have physical access to device 101, such as removing a cover or part of an enclosure, where the technician can subsequently connect equipment to a circuit board in device 101, including replacing ROM 101c. ROM 101c could
5 also comprise a nonvolatile memory, such that data is stored within ROM 101c even if no electrical power is provided to ROM 101c. In exemplary embodiments, ROM 101c could be (i) optionally omitted and storage radio unit 109 used for nonvolatile memory within device 101, or (ii) included in a memory unit within CPU 101b (not shown).

Device 101 can include a storage radio unit 109, and storage radio unit 109 can be
10 removable. Storage radio unit 109 is also depicted and described in connection with Figure 1a above and additional details regarding storage radio unit 109 are depicted and described in connection with Figures 1c and Figure 2a below. Additional details for the operation of storage radio unit 109 are provided in subsequent figures. An exemplary storage radio unit 109 form factor could comprise a standard SD card, a mini SD card, a micro SD card, a mini UICC, a
15 micro UICC, or a nano UICC, and other possibilities exist as well without departing from the scope of the present disclosure. Storage radio unit 109 could also comprise a solid state hard drive with a suitable small form factor for operation with device 101. Storage radio unit 109 can include electrical contacts (such as external pins 109x depicted in Figure 1c) which provide electrical connectivity to bus 101d.

Storage radio unit 109 can be inserted in a housing in device 101, such that storage radio unit 109 can be manually inserted into or manually removed from device 101 after device 101 is manufactured. The manual operation of insertion of storage radio unit 109 can be performed by entities in some embodiments, such as device service provider 110 or an end user of device 101. Storage radio unit 109 can include NAND or NOR flash memory in order to record data when
20 device 101 is not powered, and other nonvolatile memory technologies can be used in a storage radio unit 109 as well without departing from the scope of the present disclosure. Storage radio unit 109 can be separately manufactured from device 101 and accessed and loaded with data by a configuration unit 110d before insertion into device 101, when storage radio unit 109 includes support for operating as removable media. Storage radio unit 109 could also operate as an
25 “embedded” unit, such that storage unit comprises an integrated circuit soldered to a circuit board in device 101, and in these embodiments storage radio unit 109 can be fixed and not removable.

In exemplary embodiments, storage radio unit 109 can include a radio unit 109b and an eUICC 113 (both not shown in Figure 1b, but shown in Figure 1c), and additional details

regarding the components and operation of an eUICC 113 and radio unit 109b are depicted and described in additional figures below, including Figures 1c and Figure 2a. The inclusion of (i) a radio unit 109b supporting wireless WAN networks 102 and (ii) an eUICC 113, plus the operation of radio unit 109b with eUICC 113 in storage radio unit 109 can add functionality for storage unit 109 that is not normally included in commercially available removable storage media in the market as of 2018, such as SD cards or UICC cards.

Both radio unit 109b and eUICC 113 within storage unit 109 can include a processor, bus, and memory similar (but with less power and on a smaller scale) as the CPU 101b, bus 101d, and RAM 101e and ROM 101c. Both eUICC 113 and SRU 109 can perform cryptographic functions such as (i) internally deriving a private key, recording a private key, or deriving a symmetric key via an elliptic curve Diffie Hellman (ECDH) key exchange, (ii) recording the private key in a protected memory such that device 101 or external parties cannot feasibly or cost-effectively read the private key or derived symmetric key, and (iii) generating or verifying digital signatures using a recorded private key.

Although the exemplary environment described herein employs ROM 101c, RAM 101e, and storage radio unit 109, it should be appreciated by those skilled in the art that storage radio unit 109 could also comprise other types of computer readable media which can store data that is accessible by a device 101, such as memory cards, subscriber identity device (SIM) cards, local miniaturized hard disks, and the like, may also be used in the exemplary operating environment without departing from the scope of the disclosure. The memory and associated hardware illustrated in Figure 1b provide nonvolatile storage of computer-executable instructions, data structures, program devices, device program 101i, and other data for computer or device 101.

Note the device 101 may include a physical data connection at the physical interface 101a such as a miniaturized universal serial bus adapter, firewire, optical, or other another port and the computer executable instructions such as device program 101i, data reporting steps 101x, operating system 101h, or device driver 101g can be initially loaded into memory such as ROM 101c or storage radio unit 109 through the physical interface 101a before device 101 is given to an end user, shipped by a manufacturer to a distribution channel, or installed by a technician. Further, device program 101i, data reporting steps 101x, operating system 101h, or device driver 101g can be separately loaded into storage radio unit 109 before or after distribution of device 101, and then storage radio unit 109 can be inserted into device 101.

A number of program devices may be stored in RAM 101e, ROM 101c, or storage radio unit 109, including an operating system 101h, device driver 101g, an http client (not shown), a DNS client, and related software. Storage radio unit 109 can record program devices as well

within boot firmware, boot configuration, or operating software for storage radio unit 109, which is further depicted and described in connection with Figure 1c and Figure 2a, where the program devices in SRU 109 may be focused on storage within a nonvolatile memory 109a and a radio unit 109b. Program devices include routines, sub-routines, programs, objects,
5 components, data structures, etc., which perform particular tasks or implement particular abstract data types. Aspects of the present disclosure may be implemented in the form of (i) a device program 101i and/or data reporting libraries 101x which are executed by the device 101 working in conjunction with (ii) firmware on storage radio unit 109 (depicted as SRU firmware 109u in Figure 1c and Figure 2a) to support communications of device 101 with a wireless
10 network 102 and device service provider 110 through SRU 109 with RU 109b.

A user may enter commands and information into device 101 through an optional user interface 101j, such as a keypad, keyboard (possibly miniaturized for a mobile phone form-factor), and a pointing device. Pointing devices may include a trackball, an electronic pen, or a touch screen. A user interface 101j may also include a display such as a device screen. A
15 display may also be connected to system bus 101d via an interface. The display can comprise any type of display devices such as a liquid crystal display (LCD), a plasma display, and an organic light-emitting diode (OLED) display. Device 101 may also include a camera (not shown) connected to or integrated with device 101 through a physical interface 101a, and the camera can comprise a video camera for the device 101 to collect sensor data that includes video
20 or images. The camera (not shown) can be a CCD (charge-coupled device) camera, a CMOS (complementary metal-oxide-semiconductor) camera, or a similar device to collect video input. Other arrangements could be used as well, without departing from the disclosure.

The device 101, comprising a computer, may operate in a networked environment using logical connections to one or more remote computers, such as the servers 108 and 110a
25 illustrated in Figure 1a. Servers 108 and 110a can also function as general purpose servers to provide files, programs, disk storage, remote memory, database records, cryptographic data, and other resources to device 101 usually through a networked connection such as via wireless network 102 and IP network 111. Additional remote computers with which device 101 communicates may include another device 101 or mobile device, an M2M node within a
30 capillary network, a personal computer, other servers, a client, a router, a network PC, a peer device, a wireless network 102, an access point, or other common network node. A server 108 or 110a or a remote computer typically includes many of the elements described above relative to the device 101, including a CPU, memory, and physical interfaces. It will be appreciated that the network connections shown throughout the present disclosure are exemplary and other

means of establishing a wireless or wired communications link may be used between mobile devices, computers, servers, corresponding nodes, and similar computers.

The device program 101i and data reporting libraries 101x operating within device 101 illustrated in Figure 1b can provide computer executable instructions to hardware such as CPU 101b through a system bus 101d in order for a device 101 to (i) transmit and receive data with a device service provider 110, (ii) monitor a sensor 101f and/or change the state of an actuator 101y, (iii) send or receive data with a server B 110a, and (iv) authenticate with a server B 110a or SRU service provider 106, thus allowing server B 110a to remotely monitor or control a monitored unit 105 in an authenticated and secure manner. The device program 101i and/or data reporting steps 101x can enable the device 101 to authenticate and communicate with a server B 110b (via SRU 109 and server A 108) by recording data in memory such as RAM 101e, where the data can include sensor data, a destination IP address number such as the exemplary IP address number 110e in Figure 1a, a packet or packet header value, an encryption or ciphering algorithm and key, a digital signature and public key, etc. The data recorded in RAM 101e can be subsequently read by the operating system 101h or the device driver 101g.

The operating system 101h or the device driver 101g can write the data to a physical interface 101a comprising electrical pins 109x on SRU 109 using a system bus 101d in order to use a physical interface 101a to send application data 112a to SRU 109 and radio unit 109b and wireless network 102. Alternatively, the device program 101i and/or data reporting steps 101x can write the application data 112a directly to the SRU 109 using the system bus 101d.

The device program 101i and/or data reporting steps 101x, or operating system 101h can include steps to process the data recorded in memory such as logging data, selecting a destination address, or encoding sensor data acquired by (i) a sensor 101f or (ii) through a physical interface 101a such as a thermocouple, shock or vibration sensor, light sensor, or global positioning system (GPS) receiver, etc. The device 101 can use the physical interface 101a such as a radio to transmit or send (i) the application data 112a from a sensor or (ii) a digital signature of device 101 for application data 112a to SRU 109, where SRU 109 includes a radio unit 109b to send the application data to server A 108 via wireless network 102.

Moreover, those skilled in the art will appreciate that the present disclosure may be implemented in other computer system configurations, including hand-held devices, netbooks, portable computers, multiprocessor systems, microprocessor based or programmable consumer electronics, network personal computers, minicomputers, mainframe computers, servers, and the like. The disclosure may also be practiced in distributed computing environments, where tasks are performed by remote processing devices that are linked through a communications network.

In a distributed computing environment, program devices may be located in both local and remote memory storage devices. In addition, the terms “mobile node”, “mobile station”, “mobile device”, “M2M device”, “M2M device”, “networked sensor”, or “industrial controller” can be used to refer to device 101 as contemplated herein.

5

Figure 1c

Figure 1c is a graphical illustration of hardware, firmware, and software components for a device and a storage radio unit, in accordance with exemplary embodiments. The illustrated exemplary components for a device 101 in Figure 1c can include a device identity 101a, a device
10 clock 101m, a battery 101n, a WiFi radio 101z, a device antenna 101t, a device service provider certificate 110c, and storage radio unit 109. Device identity 101a can be a unique string or numeric identifier for device 101, such as an international mobile equipment identity (IMEI), a MAC (media access control) address, a manufactured serial number, or similar identifier. Battery 101n for device 101 can store electrical energy such that device 101 can be powered
15 when disconnected from an external power source. Clock 101m can be a clock source such as a TCXO or VCTXO to provide a timing source for (i) CPU 101b and bus 101d as well as (ii) SD card driver 101k or other device drivers 101k.

WiFi radio 101z can be a radio module supporting wireless local area network (LAN) connections via standards such as 802.11 from the IEEE. Device antenna 101t can transmit and
20 receive radio frequency signals with other nodes or devices, such as a WiFi router or other devices via WiFi direct. In exemplary embodiments, device antenna supports ultra-high frequency signals in the range of 400 MHz to 6 GHz. Device antenna 101t can have impedance that matches a radio such as 50 ohms. In some exemplary embodiments, device antenna 101t is connected to SRU 109 via electrical pins 109x, such as pins 1 and 8 as shown in Figure 2a,
25 where pins 1 and 8 are not used the 1-bit SD bus mode. Or SRU 109 could include electrical pins or connectors (separate from electrical pin 109x for an SD bus), such as with an external antenna interface 109z depicted and described in connection with Figure 2a. Although the term “SD bus” is used in the present disclosure, other bus technologies can be used for SRU 109 could include versions of a UHS bus, an SD express bus, as well as future, related versions of
30 these bus technologies for removable storage media. Further, although the term “SD card” or “micro SD card” is used for some embodiments of the present disclosure, related card technologies could be use such as cards supporting SDHC, SDXC, SDUC, and subsequent or related versions.

The illustrated components for a storage unit 109 in Figure 1c can include external pins 109x, an SD interface driver 109y, a radio unit 109b, an internal antenna 109i, a storage radio unit identity (SRU-ID) 109i, a power control unit 109e, a memory/interface processor 109d, an eUICC 113, a memory core interface 109m, an interface to an external antenna 109z, an
5 nonvolatile memory 109a, a memory controller 115, and a symmetric key 127a. The external pins 109x of storage radio unit 109 can provide an electrical interface for storage radio unit 109 to connect with a bus 101d of device 101, such that a CPU 101b of device 101 can communicate with the internal elements of storage radio unit 109, using SD card driver 101k.

Note that the use of an SD card and SD bus is not required for storage radio unit 109, and
10 other formats are possible as well, such as storage radio unit 109 comprising an integrated circuit with a PCIe bus or a SPI bus, etc. As mentioned two paragraphs above, other bus technologies for SRU 109 include a version of UHS and SD Express. An SD card can include SDXC, SDUC, SDHC, or future, related versions, and other possibilities exist as well. For the embodiments which do not use SD card formats, then “SD card driver 101k” and “SD interface
15 driver 109y” could comprise drivers appropriate for the physical form of storage radio unit 109 (such as possibly using a PCIe interface with a PCIe driver or similar). The external pins 109x can include pads or electrical contacts for electrical input into storage radio unit 109 and such as a power voltage, electrical ground, a clock input, and data lines to communicate binary data. The external pins 109x can support electrical output from storage unit 109 as well. Exemplary
20 details for pins 109x are depicted and described in connection with Figure 2a below.

In exemplary embodiments, the interface to an external antenna 109z can comprise a female receptacle for the SD card on the opposite side of the electrical pins 109x. The interface to an external antenna 109z can receive 2x male pin connectors preferably less than 0.6 mm in diameter (in order to support the height of a micro SD card), and also a pitch between the center
25 of the 2 pins of less than 2.5 mm in order to reduce space used inside the SRU 109 housing. The length of the male pins can be less than 5 mm in exemplary embodiments. Or, in other exemplary embodiments, the interface to an external antenna 109z can comprise a standard connector on the outside of the SD card housing 227 (depicted in Figure 2a below), such as the male receptacle for an u.fl antenna connection. The interface of the external antenna 109z can
30 comprise a connector both (i) mounted on the opposite side/face as pins 109x and also (ii) on the other end of a housing 227 (shown in Figure 2a below) as pins 109x.

In an exemplary embodiment, the external pins of storage unit 109 could support an SD bus or a serial peripheral interface (SPI) bus mode with a micro SD card, such that pin 6 would be ground, pin 4 would be power, pin 3 would be serial data in, pin 7 would be serial data out,

pin 5 would be a clock signal, etc. Other possibilities exist as well for the possible pin layouts and bus supported by the external pins 109x of an SD card, such as a one-bit SD bus mode or a four-bit SD bus mode. If storage unit 109 comprises a UICC, then the PIN layout of external pins 109x could comply with 3GPP standards and the interface, such as standard 3GPP TS 31.102. In another embodiment, external pins 109x can support either (i) a parallel bus such as a member of the Peripheral Component Interconnect (PCI) standards or a Peripheral Component Interconnect Express (PCIe) standard, or (ii) other serial interfaces such as a universal serial bus (USB).

Interface driver 109y within storage unit 109 can provide or be associated with the firmware to support the selected physical bus type and communications protocol for the external pins 109x. The interface driver 109y can provide functionality similar to device driver 101k or 101g in Figure 1b. For example, interface driver 109y could support one-bit or four-bit SD bus mode for data transfer between (i) the memory/interface processor 109d in SRU 109 and (ii) CPU 101b in device 101, and other possibilities exist as well. The memory/interface processor 109d can comprise a processor similar to CPU 101b, but embedded into the storage radio unit 109 and consequently potentially with less resources than a CPU 101b, such as (i) less cache memory, (ii) operating potentially at a slower speed or clock rate, (iii) fewer internal registers, (iv) lower power consumption due to reduced thermal dissipation for a storage radio unit 109 compared to device 101, etc. Memory/interface processor 109d can also manage and sequence the flow of data between non-volatile memory 109a and the external pins 109x, among other functions such as receiving updated data for radio unit 109b through external pins 109x.

As of 2018, and for the foreseeable future, large capacity removable storage media, such as SD cards or micro SD cards with a gigabyte of memory or more, can include non-volatile memory such as non-volatile memory 109a. Properly managing reading and writing to the non-volatile memory 109a can require relatively computationally intensive operations to be performed inside the removable storage media such as an SD card or storage radio unit 109. These operations performed by memory/interface processor 109d include (i) error correcting codes to reduce or eliminate bit errors in nonvolatile memory 109a, (ii) leveling write wearing in order to spread the number of data writes as uniformly as possible across the physical memory 109a, (iii) registers and tables to keep track of memory, block and page addresses in order to perform operations such as reads and writes to a memory core interface 109m, etc.

These operations can be relatively computationally intensive for a small form factor storage unit such as a micro SD card and can comprise thousands of machine code instructions for a single operation of (i) selecting a page of physical memory within a block of memory, (ii)

reading the selected page of physical memory, (iii) error correcting the data read and also write the error corrections back to the physical memory, and (iv) transferring the data through electrical pins 109x via interface driver 109y. The above exemplary operations be performed by a processor or multiple processors embedded in the storage unit 109 in the form of a memory/interface processor 109d. Exemplary microcontrollers embedded in an SRU 109 include modified versions of an 8051 processor or an ARM® processor, such as an exemplary processor that is a member of the ARM® Cortex M family of processors, and other possibilities exist as well without departing from the scope of the present disclosure. As contemplated herein, memory/interface processor 109d can be referred to as “processor 109d”, “controller 109d”, or “memory controller 109d” or “interface controller 109d”.

Secret symmetric key 127a within SRU 109 can be a secret key for use with symmetric ciphering algorithms. In exemplary embodiments, secret symmetric key 127a can be uniquely associated with SRU 109 using SRU-ID 109i and can also be loaded in storage radio unit 109 either (i) upon manufacturing of storage radio unit 109, or (ii) before distribution of storage radio unit 109 such that symmetric key 127a cannot be subsequently changed or altered after symmetric key 127a is recorded in storage unit 109. A SRU 109 can record multiple secret symmetric keys 127a, such (A) that a first key 127a remains permanently recorded in SRU 109 before installation of SRU 109 with a device 101, while (B) subsequent keys 127a could be recorded in SRU 109 after distribution of storage radio unit 109. As one example, SRU 109 could conduct an ECDH key exchange with SRU-SP 106 using (i) a private key from SRU PKI keys 109p and (ii) a public key from SRU-SP such as public key 106b corresponding to key 106a in order to derive a subsequent, second key 127a for use with encrypting and decrypting application data with SRU-SP 106. In other words, these subsequent symmetric keys 127a could be changed or altered after the first symmetric key 127a was written before distribution of storage unit 109.

Symmetric key 127a or 127b in Figure 1c could comprise a pseudo random number or random number of sufficient length such as an exemplary 128 or 256 bits, such that output of a symmetric ciphering algorithm such as the Advanced Encryption Standard (AES) could not feasibly be decrypted without the use of a symmetric key 127a or 127b. Among many possible uses for symmetric key 127a in the present disclosure, symmetric key 127a could be utilized to decrypt SRU unit firmware 109u, where symmetric key 127a was used to encrypt the SRU firmware 109u before SRU firmware 109u was loaded into storage unit 109. In this manner, SRU unit firmware 109u can be resistant to reading or tampering by end users or other external parties other than the manufacturer of SRU 109 or authorized parties who also have symmetric

key 127a. A symmetric key 127a could also be derived from the network access credentials, such as a key K associated with an IMSI or SUPI, in order to encrypt application data 112a transferred to Server A 108.

Storage radio unit 109 can also include a memory core interface 109m, where memory core interface 109m provides processor 109d and/or radio unit 109b access to physical memory such as non-volatile memory 101a and/or RAM 213 (not shown in Figure 1c but depicted and described in connection with Figure 2a below). Controller 109d and/or radio unit 109b can read data from physical memory by writing an address of a memory page and/or block to memory core interface 109m, and memory core interface 109m returns the data from the page in physical nonvolatile memory 109a or RAM 213.

Similarly, memory processor 109d and/or radio unit 109b can write data from physical memory by writing an address of a memory block and/or page to memory core interface 109m plus the data to be recorded in physical memory, and memory core interface 109m can subsequently write the data to physical memory at the block and page address specified. Other possibilities exist as well for the use of a memory core interface 109m without departing from the scope of the present disclosure. In exemplary embodiments, individual cells within a page can be addressed by a memory core interface 109m as well, and other possibilities for the structure and naming conventions of memory are possible without departing from the scope of the present disclosure.

Storage radio unit 109 can include radio unit 109b, where radio unit 109b was also depicted and described in connection with Figure 1a above, and additional details regarding radio unit 109b are depicted and described in connection with Figure 2a below. Radio unit 109b can include radio processor 109c for controlling radio unit 109b, where processor 109b can be similar to (a) CPU 101b for device 101 from Figure 1b, or (b) memory/interface processor 109d described above. Radio processor 109c can be referred to herein as “processor 109c”. The processor 109c in radio unit 109b can function similar to processor 109d for SRU 109 as described above, with a form factor, speed, and computational capacity suitable for radio unit 109b. Processor 109c in radio unit 109b could be a processor belonging to the exemplary MIPS®, ARM®, or Intel® family of processors, and other possibilities exist as well.

Symmetric key 127b can be equivalent to symmetric key 127a for SRU 109 described above. In exemplary embodiments, a first symmetric key 127b can be used with a TLS-PSK session with SM-SR 107, and in this embodiment the first symmetric key 127b can be recorded in eUICC 113. In addition, a second symmetric key 127b can be used with a DTLS-PSK session with server A 108, where application data 112a is encrypted with the second symmetric key

127g and a symmetric ciphering algorithms, and server B response 112b data is decrypted with the second symmetric key 127b and the symmetric ciphering algorithm. Thus, a radio unit 109b can also operate with multiple symmetric keys 127b, and the multiple pre-shared secret symmetric keys 127b can also include or each be associated with a MAC key, where a MAC key is depicted and described in connection with Figure 4b below. A symmetric key 127b can also separately be used to encrypt and decrypt data for radio unit 109b which is stored in memory 109r.

Processor 109c can include components such as registers, accumulators, and logic elements to add, subtract, multiply, and divide numerical values, and processor 109c can be connected to an internal bus 212 for SRU 109 depicted in Figure 2a below. The timing of processor 109c by an internal clock for SRU 109 as RF clock 205 depicted in Figure 2a below. Note that in some exemplary embodiments, the timing of processor 109c is not driven by device clock 101m (which can be less precise than requirements for NB-IoT communications 104 in Figure 1a), and rather processor 109c is driven by a separate, internal clock 205 within SRU 109. Processor 109c can provide (i) the hardware for radio unit 109b to perform calculations for radio communications with wireless network 102 in addition to (ii) the general operation of radio unit 109b and managing communication between radio unit 109b and other components within SRU 109 via an internal bus 212 of SRU 109 (shown in Figure 2a below). Although processor 109c is depicted in Figure 1c as internal to radio unit 109b, processor 109c could be external to radio unit 109b. Further, in some exemplary embodiments, processor 109d and processor 109c could be combined into a single processor. For these embodiments, the combined processor could have the functionality of the separate processors 109d and 109c depicted in Figure 1c.

The storage radio unit identity (SRU-ID) 109i can comprise a globally unique identifier for SRU 109, such that nodes communicating with device 101 using storage radio unit 109 can properly identify SRU 109 using the SRU-ID 109i. SRU ID 109i can be a string or a number, such as a hexadecimal number with sufficient length such as an exemplary 8 bytes of data for the SRU-ID 109i, although other possibilities exist as well without departing from the scope of the present disclosure. In exemplary embodiments, SRU-ID 109i can be written to hardware such as an EEPROM in SRU 109 upon manufacturing of SRU 109 or configuration by an SRU-SP 106 and SRU-ID 109i is not normally modified or changed after distribution of SRU 109 from an SRU-SP 106 or receipt of SRU 109 by a user. Other possibilities exist as well for the format, recording location, and use of a SRU-ID 109i without departing from the scope of the present disclosure. In exemplary embodiments, SRU-ID 109i can be written to protected

memory or a hardware address within SRU 109 such that a device 101 could not feasibly modify or change the SRU-ID 109i.

eUICC 113 in SRU 109 can comprise an embedded universal integrated circuit card supporting standards such as GSMA SGP.02 or GSMA SGP.22, including subsequent or related versions of these standards. For embodiments where eUICC 113 supports standard SGP.22, then device 101 could operate a local profile assistant (LPA) as an application in device programs 101i, and the eUICC interface family ES10 from SGP.22 could be passed to eUICC 113 via electrical pins 109x. eUICC 113 was also depicted and described above in connection with Figure 1a. eUICC 113 can include an eUICC ID 113a in order to unique identify eUICC 113 across a plurality of globally distributed eUICCs. Although not depicted in Figure 1c, eUICC 113 can record a set of ECC PKI keys for both eUICC and at least one certificate authority. In exemplary embodiments, eUICC 113 records a root certificate for the GSMA for eUICC operations, which eUICC 113 can use to authenticate an SM-DP. eUICC 113 includes a set of cryptographic algorithms for conducting ECDH key exchanges, symmetric ciphering, and digital signature algorithms using elliptic curve cryptography. eUICC 113 can establish a TLS session with SM-SR 107 using a pre-shared secret key (PSK), which could comprise a symmetric ciphering key 127b recorded within eUICC 113 and also be associated with SM-SR configuration data 113b.

As depicted in Figure 1c, eUICC 113 also records SM-SR configuration data 113b, which could include an IP address, domain name, and domain name service (DNS) server IP address in order for the eUICC 113 to resolve a name for SM-SR 107 into an IP address equivalent to IP address 110e depicted in Figure 1a. SM-SR configuration data 113b could also include a PSK for use with a TLS session with SM-SR 107, where the PSK can be a symmetric ciphering key 127b and an associated MAC key (where a MAC key is depicted and described in connection with Figure 4b). eUICC 113 can communicate internally with radio unit 109b using an internal bus 212 as depicted and described in connection with Figure 2a below.

eUICC 113 can record a bootstrap profile 113c, which can include information equivalent to that recorded in a traditional SIM card for connection to a first wireless network 102. eUICC 113 can record an operational profile 113d, which can include information equivalent to that recorded in a traditional SIM card for connection to a second wireless network 102. Or, the bootstrap profile 113c could be used by RU 109b to connect with the second wireless network 102 in a “roaming” mode, and the operational profile 113c could be used by RU 109b to connect with the second wireless network 102 in a “native” mode or with credentials or key K also recorded by the second wireless network 102. The bootstrap profile

113c can be used in a startup or “initial” or “failover” state for eUICC 113 and radio unit 109b, such that connectivity to wireless network 102 could be obtained if a working operational profile 113d is not available for eUICC 113. Operational profile 113d can be used by eUICC 113 and radio unit 109b for preferred operation of device 101 within a wireless network 102.

5 Although illustrated as a separate unit from radio unit 109b in Figure 1c, in exemplary embodiments, eUICC 113 could also be optionally combined with radio unit 109b or processor 109c (where processor 109c could also be combined with processor 109d). In other words, a single physical processor similar to CPU 101b could perform the function of radio processor 109c and eUICC 113. eUICC 113 could comprise also a trusted execution environment (TEE),
10 “secure processing environment”, “smart secure platform” or similar secured environment within a larger, more generalized processor such as processor 109c.

Non-volatile memory 109a in storage radio unit 109 can include device and SRU accessible nonvolatile memory 109f and SRU only accessible nonvolatile memory 109r. The two types of memory can be identified by separate addresses within the physical memory
15 comprising non-volatile memory 109f, or similarly different sectors assigned by a file system written to the physical nonvolatile memory. Further, memory 109f and 109r could represent different partitions within a memory 109a, where memory 109a has been formatted with a file system. In an exemplary embodiment the two types of memory 109f and 109r can be segmented logically, where memory 109r is encrypted with a symmetric key 127a within SRU 109 by a
20 memory controller 115, where memory controller 115 is also depicted and described in connection with Figure 2a below.

Device and SRU accessible memory 109f can include shared nonvolatile memory 109h and device general memory 109g. Device general memory 109g can comprise memory that device 101 can access in non-volatile memory 109a such as recording firmware for device 101
25 or other long-term and non-volatile storage of data or files for device 101. In exemplary embodiments, operating system 101h for device 101 can be recorded in device general memory 109g. Device and SRU accessible memory 109f can record a file system for both device 101 and SRU 109 to access, such as exemplary file systems of FAT16, FAT 32, NTFS, ext3, ext4, UDF, and other file systems for memory 109g and memory 109h are possible as well without
30 departing from the scope of the present disclosure.

As contemplated herein, memory 109f can represent physical memory cells within a nonvolatile memory 109a, and memory 109f and memory 109r can represent separate file systems or logical and secure separation of memory 109f and 109r. In exemplary embodiments, a file system within memory 109f can be specified as compatible with a file system for OS 101h

for device 101, while a file system within memory 109r can be a different file system and file system type preferred for internal use by SRU 109. For an exemplary embodiment, the file system type for memory 109f can comprise a FAT32 or exFAT file system, and memory 109r can comprise a JFFS2 or ext2 file system. Other possibilities exist as well regarding the use of a file system for memory 109f and memory 109r, without departing from the scope of the present disclosure. Additional details regarding the format, structure, and files within nonvolatile memory 109a and 109h will be provided below in connection with Figures 2a, 2b, 2e, and 2f, as well as other descriptions herein.

Shared memory 109h in nonvolatile memory 109f can include memory accessible by both a device 101 and SRU 109, such as a portion of a file system recorded in memory 109f. Exemplary use of device and SRU shared memory 109h is depicted and described in connection with Figure 2a below. In exemplary embodiments, both device 101 and SRU 109 can both read and write to device and SRU shared nonvolatile memory 109h, which can also be referenced herein as shared nonvolatile memory 109h or shared memory 109h. As illustrated in Figure 1c, shared memory 109h can be a subset of device and SRU accessible memory 109f, and device and SRU accessible memory 109f can also include general memory 109g for device 101. Note that (i) RU 109b can read and write directly to shared memory 109h, or (ii) RU 109b can read and write to shared memory 109h through a memory processor 109d in SRU 109. In other words, for (ii) RU 109b could communicate to memory processor 109d via an internal bus, and processor 109d could read and write to memory 109h on behalf of RU 109b.

In exemplary embodiments, the SRU 109 may have access to general memory 101g, such as to support commands or instructions from SRU-SP 106, or device 101, or device service provider 110, which could comprise instructions to fix memory, reallocate file space or structure, or other system administration operations. However, without outside instruction to SRU 109 from device 101 or SRU-SP 106 or device service provider 110, in regular operating modes SRU 109 may preferably not use memory 109g for the internal operation of SRU 109, where memory 109g can comprise memory and files for general nonvolatile storage of data for device 101. In exemplary embodiments, memory 109h can comprise a first directory within a file system for memory 109f and memory 109g for device 101 can comprise a second directory within a file system for memory 109f. Other possibilities for the configuration of a non-volatile memory 109a are possible as well, such that storage radio unit 109 can provide nonvolatile memory (i) dedicated to device 101, (ii) dedicated to SRU 109, and (iii) shared between device 101 and SRU 109 for passing data between device 101 and RU 109b, without departing from the scope of the present disclosure.

Data utilized by radio unit 109b may logically or physically separated for use by radio unit 109b and processor 109c or processor 109d through the use a memory controller 115. The data used by radio unit 109b and processors 109c and 109d can be recorded in SRU only accessible nonvolatile memory 109r. As contemplated herein, SRU only accessible nonvolatile memory 109r can be also be referred to as “SRU memory 109r”, “SRU 109 protected memory 109r”, “SRU accessible memory 109r”, or “protected memory 109r”.

Memory controller 115 can provide physical or logical isolation of SRU 109 only accessible memory 109r such that device 101 could not read or write to SRU only accessible memory 109r. Memory controller 115 could prevent device 101 from reading or writing data to SRU 109 only accessible memory 109r. As one example, the physical memory in storage unit 109 comprising non-volatile memory 109a could have a portion of the physical addresses reserved for reading and writing by SRU 109 only, such as an exemplary top 50 blocks of non-volatile memory 109a, and other values are possible as well. Memory controller 115 could intentionally disallow the transfer of data from or to non-volatile memory 109r with external bus 202, where the block address is in the exemplary top 50 blocks.

Memory controller 115 could also operate on a lower level than a block address for non-volatile memory 109a as well, such as only allowing SRU 109 or processor 109c and processor 109d to allow a specified range of pages within non-volatile memory 109a, where the specified range of pages could belong to SRU only accessible memory 109r. In this manner, memory controller 115 could operate as a firewall to restrict access to SRU only accessible memory 109r. Other possibilities exist as well for the operation of a memory controller 115 in order to isolate and separate SRU only accessible memory 109r such that interface controller 109d or device 101 cannot utilize physical memory addresses for recording and reading data utilized in a RU only accessible memory 109g.

In another embodiment, memory controller 115 could perform hardware-based encryption/decryption using (i) a symmetric key 127b to encrypt and decrypt data transferred between RU 109b and SRU only accessible memory 109r, or (ii) symmetric key 127a to encrypt and decrypt data transferred between processor 109d and SRU memory 109r. In an exemplary embodiment, SRU only accessible memory 109r can be formatted with a file system with a separate partition from the memory that device 101 accesses within memory 109f. A file system on SRU only accessible memory 109r could be encrypted using a symmetric ciphering algorithm (from cryptographic algorithms 109q) and (i) a symmetric key 127b recorded by RU 109b or (ii) symmetric key 127a used by processor 109d, such that even if RU only accessible memory 109r could be accessed by device 101, no useful data could be extracted or tampered with (since

symmetric keys 127a and 127b could also include MAC keys for data integrity). Although memory controller 115 is depicted in Figure 1c as a separate element from memory core interface 109m, memory controller 115 could also be optionally integrated with memory core interface 109m, and other possibilities exist as well for the operation and location of a memory controller 115 for ciphering data or restricting access to data recorded in storage unit 109 for SRU 109 with protected memory 109r, without departing from the scope of the present disclosure.

As depicted in Figure 1c, SRU memory 109r can include a radio unit AT control stack 109s, SRU firmware 109u, PKI keys 109v, and cryptographic algorithms 109q. Other data for the operation of SRU 109 and radio unit 109b can be recorded in SRU memory 109r as well, such as the exemplary data depicted and described below for protected memory 109r in Figure 2a. AT control stack 109s can comprise software and firmware for SRU 109 and processor 109d to operate as an AT client for radio unit 109b, where processor 109c operates as an AT host for radio unit 109b. As of 2018, most commercially available radio modules for devices such as an RU 109b may be driven or operated with an AT command set.

An SRU 109 could include a commercially available radio unit 109b or similarly a module (possibly as a wafer level chip scale package) in order to fit within a micro SD card or SRU 109 with a housing 227 from Figure 2a below. A commercially available module for RU 109b may require AT command sets such as those specified by 3GPP in TS 27.007. Consequently, in order to utilize a commercially available module for radio unit 109b, SRU 109 could record radio unit AT control stack 109s. For embodiments where AT command controls are not utilized by radio unit 109b, then SRU 109 could record and use the client-side commands and instructions in stack 109s.

SRU firmware 109u can comprise the firmware, software, operating system, and various libraries or packages for SRU 109 to operate after a boot process. Additional details regarding SRU firmware 109u are provided in Figure 2e below. PKI keys 109v can record private keys for use by SRU 109 and RU 109b, such as a SRU private key SRU SK 109p from Figure 1a and public keys for SRU-SP 106 such as SRU-SP PK 106b. PKI keys 109v can include multiple PKI keys, such as a public key for SRU-SP 106 to use with code verification or signatures on SRU firmware 109u. In other words, SRU-SP 106 could provide or record the firmware 109u with a digital signature, and SRU 109 could verify the digital signature using a public key from SRU-SP 106 in memory 109r. PKI keys 109v could also record ephemeral PKI keys derived by SRU 109 or RU 109b. PKI keys 109v could also record a public key for SRU 109 for use by device 101 or SRU-SP 106 or device service provider 110, such as SRU public key SRU PK

109w, and SRU PK 109w could be recorded with a certificate for SRU 109. PKI keys 109v can also record a set of cryptographic parameters 191e or cryptographic settings 192e (depicted in Figure 2b below and later figures as well, such as Figure 4b).

In exemplary embodiments, initial PKI keys for SRU 109 (such as keys 109p and 106b, as well as associated certificates and certificate issuer root certificates) may be recorded in EEPROM as shown in Figure 2a, and subsequent or updated versions of the keys can be recorded in memory 109r. PKI keys 109v can also record a set of root certificates for SRU 109 to use with verification of certificate authority digital signatures for certificates such as certificate 110c. Root certificates recorded in PKI keys 109v can comprise the list or a subset of the list of included root certificates from the Mozilla Foundation with Mozilla projects, where the aggregate list of community approved root certificates and associated parameters is in the widely distributed file certdata.txt from the Mozilla web site.

Cryptographic algorithms 109q in memory 109r can comprise digital signature algorithms such as ECDSA or DSA, symmetric ciphering algorithms such as AES and TripleDES, key exchange algorithms, key pair generation algorithms, random number generators, secure hash algorithms such as SHA-2 or SHA-3, and asymmetric ciphering algorithms such as El gamal or RSA. Libraries in cryptographic algorithms 109q could be from standard open source libraries such as OpenSSL, crypto++, Bouncy Castle, or Mozilla libraries from Network Security Services, and other possibilities exist as well.

Figure 2a

Figure 2a is a graphical illustration of components within a storage radio unit, in accordance with exemplary embodiments. Figure 2a depicts additional details for a radio unit 109b and storage radio unit 109 illustrated in Figure 1c. A system 200a can comprise a SRU 109 and an external antenna 201i. Note that external antenna 201i could be within device 101 and could also be the same as a device antenna 101t. Storage radio unit 109 can include nonvolatile storage memory 109a, radio unit 109b, eUICC 113, phased loop logic (PLL) 211, memory core interface 109m, memory/interface processor 109d, radio processor 109c, external data bus 202, an internal bus 212, electrical pins 109x, power detector 209, power capacitor 208, RAM memory 213, memory controller 115, EEPROM 216, internal antenna 109t, interface to external antenna 109z, and SD driver 109y, and power control unit 109e. Storage radio unit 109 can be enclosed in an SRU housing 227, where SRU housing 227 can be compatible with standard physical housings such as a standard SD card, a micro SD card, a mini UICC, or possibly an integrated circuit package. Figure 2a also depicts an external antenna 201i which

can be connected to the interface to external antenna 109z. Although radio processor 109c is depicted external to radio unit 109b in Figure 2a, radio processor 109c could also comprise a logical or physical component in radio unit 109b. Or, the function of radio processor 109c and memory/interface processor 109d could be combined into a single processor.

5 Electrical pins 109x are depicted as the pins for a micro SD card supporting 1 bit SD bus mode, although other possibilities exist as well for the physical electrical connections for SRU 109. Electrical pins 109x can comprise pin 1 201a and pin 8 201b for connection to a device antenna such as device antenna 101t in Figure 1c, pin 2 201c for chip select (CS), pin 3 201d for data input (MOSI) to SRU 109, pin 5 201e clock input from an clock in device 101 such as
10 clock 101m, pin 6 201f for data output (MISO) from SRU 109, pin 7 201g for electrical ground, and pin 4 201h for powering voltage input (which ranges from 2.7 to 3.6 volts for the micro SD card format).

 Electrical pins 109x can be connected to an external bus 202, which can be connected to memory/interface processor 109d. Electrical pins 109x do not need to be electrically combined
15 into a single circuit, and the separate pins, traces, or wiring lines can be separately connected to processor 109d, and the external facing pins can collectively be considered an external bus 202. In exemplary embodiments, an SD card format and electrical pin connections are not required, and other physical forms of electrical connections could be supported as well, such as PCIe, SPI, serial, or USBv3. Note that pins 1 201a and pin 8 201b can be used as signaling lines for
20 various SD bus technologies, and not with a device antenna for some embodiments, such as embodiments with the use of an external antenna 201i.

 As depicted in Figure 2a, radio unit 109b can include an RF front end 203, an RF transceiver 204, an RF clock 205, a baseband unit 206, a baseband clock 207, and an eUICC 113. Radio unit 109b can be connected to internal bus 212 in order to send and receive electrical
25 signals with both radio processor 109c and memory/interface processor 109d, as well as the other electrical components connected to internal bus 212. As noted above, RU processor 109c could also operate within RU 109b. RF front end 203 can include SAW filters, a power amplifier for a transmit chain, a low noise amplifier for a receive chain, a switching circuit for to switch between transmit and receive in half-duplex mode, and logic and circuits to convert radio
30 frequency signals between an antenna and an intermediate frequency for processing by the RF transceiver 204. RF front end 203 can include an oscillator, phased loop logic, and a mixer and supporting components.

 In exemplary embodiments, RF front end 203 includes an antenna switch 203a. Antenna switch 203a can provide connectivity to each of device antenna 101t through pin 1 201a and pin

8 201b, the interface to external antenna 109z, and internal antenna 109t. In an exemplary embodiment, switch 203a can be selected or set to use one of the three antennas by radio processor 109c using general purpose input/output (GPIO) lines, or similar signals from radio processor 109c. In another exemplary embodiment, switch 203a can automatically determine
5 which of the three antennas (101t in device 101, 109t, or 201i) to use with RF front end 203 based on detecting a 50 ohm impedance along the three possible connections (e.g. pins 1,8 of 109x, through 109z, or with 109t). In exemplary embodiments where RU 109b detects a matching impedance for at least two of the exemplary three antennas depicted (e.g. 101t in device 101 in Figure 1a, 109t, or 201i), then RU 109b could use multiple in and multiple out
10 (MIMO) radio modulation.

RF transceiver 204 can include electrical components to convert between (a) intermediate frequencies with RF front end 203 and (b) baseband frequencies and signals with baseband unit 206. RF transceiver 204 can include digital to analog converters, analog to digital converters, and also use input from RF clock 205 for timing signals. RF transceiver 204 can
15 implement (i) OFDM and QPSK signal modulation and (i) separate RF channels with multiple sub-carriers required by wireless network 102, if wireless network supports LTE and 5G and related standards. High order modulation could be supported such as 16-QAM, 64-QAM, 256-QAM, etc. For low bandwidth applications, BSPK or QPSK could be used.

In exemplary embodiments, RF clock 205 frequency is selected to match or support the
20 wireless WAN networking protocol, such as using a RF clock frequency of 38.4 MHz for LTE NB-IoT applications, although other frequencies could be utilized as well, such as, but not limited to, 19.4 Mhz. Note that SRU 109 may need a separate, internal clock comprising RF clock 205 than clock 101m and input clock via 201e, since that clock source for SD cards or an SD bus or similar PCI or SPI busses is normally not sufficiently accurate or stable for a radio
25 unit 109b. RF transceiver 204 can implement logic and algorithms for an intermediate frequency in order to support the protocol used by wireless network 102, which could be NB-IoT, 5G, and other possibilities exist as well.

Baseband unit 206 can include electrical components to convert between baseband frequency signals communicated with RF transceiver 204 and data with radio processor 109c.
30 Baseband unit 206 can implement radio control functions such as signal modulation, encoding, radio frequency shifting, carrier selection, frequency tuning. For NB-IOT 104 networks, baseband unit 206 can implement baseband physical channels, such as processing narrowband primary synchronization signal (NPSS) and narrowband secondary synchronization signal

(NSSS) to synchronize its frequency and timing with an eNB. Radio unit 109b can adjust or offset clock input from clock 205 in order to synchronize timing with an eNB.

After RU 109b synchronizes timing with wireless network 102, baseband unit 206 by receiving data from RF transceiver 204 can process and read data from a narrowband broadcast control channel (NPBCH). Narrowband physical random access channel (NPRACH) can be implemented by baseband unit 206 to establish initial association with wireless network 102. Baseband unit 206 can structure downlink and uplink data packets with a narrowband physical downlink shared channel (NPDSCH) and narrowband physical uplink shared channel (NPUSCH), respectively. Although baseband unit 206 is depicted as separate from radio processor 109c and RF transceiver 204, baseband unit 206 could be combined with either radio processor 109c and/or RF transceiver 204. In exemplary embodiments, baseband unit 206 can be driven by a baseband clock 207, which can operate at a different and lower frequency than RF clock 205. In some exemplary embodiments, baseband clock 207 operating at 32.768 kHz, although other possibilities exist as well.

As depicted in Figure 2a, eUICC 113 can also be implemented within radio unit 109b, and connected to a processor 109c operating in radio unit 109b via an internal bus for radio unit 109b. Or, eUICC 113 can be implemented as external to radio unit 109b and connected to radio unit 109b and processor 109c via internal bus 212 for SRU 109 (which is depicted in Figure 1c). PLL 211 operating in SRU 109 can comprise a timing circuit to convert RF clock 205 into a frequency to drive processors 109c and/or 109d. In an exemplary embodiment, radio transceiver 204 can be driven by a RF clock 205 with a frequency in a range of less than 50 MHz, such as an exemplary 38.4 or 19.2 MHz, while processors 109c and 109d can preferably be driven at a higher frequency such as a frequency greater than 100 Mhz.

PLL 211 can step up or convert the frequency of RF clock 205 into a higher value or frequency. In an exemplary embodiment where RF clock 205 operates at 38.4 MHz, PLL 211 can multiple the frequency of RF clock 205 to a higher value of 5x times, and thus output a frequency of 192 MHz to drive processor 109c and processor 109d. PLL 211 could use different multiplying RF clock value 205 other than the exemplary value of 5, such as exemplary integer values in the range of 3x to 8x, and other possibilities exist as well. Note that the output of PLL 211 can also be input to eUICC 113, since eUICC 113 can operate with an internal processor in order to perform cryptographic operations. PLL 211 can implement separate multiplication values of RF clock 205 in order to drive each of processor 109c, processor 109d, and eUICC 113 with different clock input values, using RF clock 205 as a source. PLL 211 can function as an

oscillator (i) receiving input from RF clock 205, and (ii) outputting a CMOS timing signal at 100 megahertz or higher.

Random access memory 213 in SRU 109 can function similar to RAM 101e for device 101, with a form factor, speed, and storage capacity suitable for SRU 109. RAM 213 can be connected to internal data bus 212 in storage radio unit 109, and can store data for processors 109c and 109d in a volatile state, such that data recorded in RAM 213 can be essentially flushed or reset when electrical power to storage radio unit 109 and radio unit 109b is removed. Random access memory 213 can store data such as tables of memory addresses and sectors for memory 109f and memory 109r, since these tables are ordinarily larger than the registers provided by processors 109c and 109d. RAM 213 can be shared or allocated between processors 109c and 109d, or RAM 213 can comprise two separate RAM units, where one RAM memory unit is allocated to processor 109c and a second RAM memory unit is allocated to processor 109d.

In addition, for some exemplary embodiments, RAM 213 can comprise a nonvolatile RAM (NVRAM) as described in connection with Figure 5a and depicted for a RAM memory 101e' in Figure 8. Since SRU 109 may be powered off frequently and at times not controlled by SRU 109 (such as under the control of device 101), for some exemplary embodiments RAM 213 comprises an NVRAM. For these embodiments, the time required for a boot process for SRU 109 and RU 109b can be reduced (since a last operating state can be stored in RAM 213 as NVRAM), and thus power or energy consumption by SRU 109 can be reduced.

EEPROM 216 in SRU 109 can include SRU ID 109i, a certificate 106d for SRU-SP 106 with an initial SRU-SP public key SRU-SP PK 106b corresponding to the SRU-SP private key SRU-SP SK 106a, symmetric keys 127, SRU boot firmware 217, SRU boot configuration 218, and PKI keys 109v' for SRU 109. Symmetric keys 127 can comprise a collection of pre-shared secret keys 127a and 127b. EEPROM 216 could also comprise a read only memory, where the data in EEPROM 216 is written once upon manufacturing of storage unit 109. EEPROM 216 could also function as a read only memory for SRU 109 similar to ROM 101c for device 101 above. In other words, EEPROM 216 does not need to be erasable and reprogrammable, although some data in EEPROM 216 could be re-written in exemplary embodiments.

Data within EEPROM 216 can comprise SRU boot firmware 217 and CU boot configuration 218. Upon startup or powering of storage radio unit 109, processor 109d could load and operate on instructions provided by boot firmware 217 and boot configuration 218 from EEPROM 216. Boot firmware 217 and boot configuration 218 can provide instructions to processor 109d in the format of machine executable code. The instructions in boot firmware 217 can provide instructions for processor 109d and possibly processor 109c to initialize and

subsequently load SRU firmware 101u from memory 109r via internal bus 212. Data or files can be written to EEPROM 216 by a configuration unit similar to configuration unit 110d, where the configuration unit for EEPROM 216 can be operated by (i) the manufacturer of SRU 109 or (ii) SRU-SP 106. The data in EEPROM 216 can be written into EEPROM 216 in SRU 109 before SRU 109 is distributed to a device service provider 110, an SRU distributor, or an end user of device 101, such as during a step 260 depicted and described in connection with Figure 2f below.

PKI keys 109v' in EEPROM 216 can comprise an initial set of PKI keys 109v depicted and described in connection with Figure 1c, with a difference being (a) PKI keys 109v' could be recorded in a read-only state upon manufacturing of SRU 109, and (b) PKI keys 109v in nonvolatile memory 109a could comprise a subsequently updated version of the initial set of PKI keys 109v'. PKI keys 109v' can include a certificate for SRU 109. A certificate for SRU 109 in PKI keys 109v' can include the SRU identity 109i, SRU public key SRU PK 109w for SRP private key SRU SK 101p, certificate parameters, and a certificate authority digital signature.

SRU certificate for SRU 109 (and other certificates herein) can be formatted according to the X.509 v3 specifications, among other possible formats, and stored as a plain text file, *.pem file, or *.crt file or similar file formats. SRU certificate in PKI keys 109v' can be used by SRU 109 in order to (i) verify identity of SRU 109 to device 101 or SRU-SP 106. SRU private key 109p in PKI keys 109s can be used to generate a digital signature. In exemplary embodiments, parameters in a SRU certificate in PKI keys can include an expiration time of longer than the expected operational lifetime of storage radio unit 109, and in this manner SRU certificate in PKI keys 109s can remain valid whenever storage unit 109 is utilized. An exemplary expiration time of SRU certificate in PKI keys 109s could be 20 years, although other possibilities exist as well. For embodiments where a certificate for SRU 109 is updated (such as changing a set of cryptographic parameters 191e), then the updated certificate for SRUP 109 could be stored or recorded in PKI keys 109v in protected memory 109r and the original certificate could continue to be recorded in EEPROM 216.

SRU boot firmware 217 in EEPROM 216 can provide machine executable code for processor 109d to initiate operations when electrical power is provided to SRU 109 via the electrical pins 109x. Although not illustrated in Figure 2a, processor 109d may also include a ROM memory with processor 109d instructions for processor 109d to fetch boot firmware 217 and boot configuration 218 upon startup when power is provided to processor 109d. SRU boot firmware 217 can include memory control libraries 217a. SRU boot firmware 217 in EEPROM

216 could also include instructions for SRU 109 to load SRU firmware 109u recorded in non volatile memory 109a, if present, as depicted in Figure 2a.

In exemplary embodiments, SRU firmware 109 could be (i) recorded in memory 109r as encrypted with symmetric key 127a, and (ii) decrypted with a symmetric key 127a recorded in EEPROM 216 within symmetric keys 127. Authorized providers of SRU firmware 109u, such as SRU-SP 106 or device service provider 110, could have access to symmetric key 127a for SRU 109 with SRU ID 109i, and consequently only the authorized providers could properly encrypt SRU firmware 109u using the symmetric key 127a. In exemplary embodiments, symmetric key 127a is unique for each SRU 109 and associated with each unique SRU-ID 109i. SRU firmware 109u could be transmitted to a device 101 over network 102 in the encrypted format and encrypted with the symmetric key 127a for SRU 109.

Or, SRU firmware 109u can be written to SRU 109 by a configuration unit similar to configuration unit 110d and operated by SRU-SP 106 or device service provider 110 or a manufacturer for SRU 109, and other possibilities exist as well for the source and recording of an initial SRP firmware 109u for SRU 109 before SRU 109 is inserted in device 101. Symmetric key 127a within EEPROM 216 can comprise a shared secret key. Symmetric key 127b for SRU 109 can comprise a different key for other purposes than decrypting SRU firmware 109u. A configuration unit 110d could write a symmetric key 127b to EEPROM 216 or protected memory 109r, where the symmetric key 127b could be used to encrypt application data 112a and decrypt a response 112b. In some exemplary embodiments, SRU firmware 109u is not required for SRU 109 to operate, and SRU 109 could operate using SRU boot firmware 217. SRU firmware 109u can be loaded into storage unit 109 for embodiments where the firmware for SRU 109 is desired to be updated.

Memory control libraries 217a could include software or firmware to manage and schedule the operation of SRU 109, such as machine code or machine executable instructions for (i) instructing processor 109d to write data to bus 212 for memory controller 115 when data is recorded in memory 109r (e.g. memory write libraries 217b), (ii) read data from external bus 202 when data from device 101 is passed to SRU 109 (e.g. memory read libraries 217c), and (iii) reading SRU firmware 109u from protected memory 109r for SRU 109. For embodiments where processor 109c and processor 109d are combined, memory control libraries 217a can include the software libraries and firmware for the combined processor to manage all input and output of SRU 109 upon boot. Other possibilities exist as well for memory control libraries 217a to support the operation of SRU 109 via program instructions provided to processor 109d without departing from the scope of the present disclosure. In exemplary embodiments, boot

firmware 217 and boot configuration 218 can provide boot instructions and setup for both processor 109c and processor 109d.

SRU only accessible memory 109r within nonvolatile memory 109a could be accessed by SRU 109 and processors 109d and 109c via an internal bus 212 and memory controller 115.

5 SRU only accessible memory 109r within nonvolatile memory 109a can comprise a protected memory (such that memory 109r cannot be read by device 101 or written to by device 101). As contemplated herein, SRU only accessible memory 109r can also be referred to as protected memory 109r. Internal bus 212 and memory controller 115 can be utilized to physically or logically separate protected memory 109r from memory 109f and memory 109g in Figure 1c, in
10 order to prevent device 101 or external parties with physical access to storage unit 109 from reading the data or writing data in protected memory 109r. In other words, and as illustrated in Figure 2a, internal bus 212 and memory controller 115 can limit (A) the ability to read and write data to protected memory 109r only to (B) SRU 109 in exemplary embodiments. In contrast, access by device 101 to memory 109f could be accomplished through the separate data bus 202
15 and interface controller 109d.

In exemplary embodiments, memory processor 109d can provide the functionality to device 101 for storage radio unit 109 to operate as a standard flash memory storage unit via data bus 202. Internal bus 212 and memory controller 115 can provide memory resources to SRU 109 in a manner that prevents device 101 from accessing protected memory 109r, including the
20 description of memory controller 115 above. Internal bus 212 is not required in some exemplary embodiments, and memory controller 115 could be connected to data bus 202, but limit access to physical memory cells comprising memory 109r to processors 109d and 109c. Other possibilities for (i) limiting access to memory 109r to SRU 109, while (ii) supporting shared access to memory 109h with device 101 are possible as well without departing from the scope of
25 the present disclosure. In an exemplary embodiment, memory 101f, and 109r could represent separate partitions for nonvolatile memory 109a.

SRU only accessible memory 109r in nonvolatile memory 109a can include data for the operation of SRU 109 which would not normally be read or utilized by device 101. In addition to the exemplary data depicted and described for protected memory 109r in Figure 1c, memory
30 109r can include an operating system (OS) 220 for SRU 109, cryptographic algorithms 109q, device service provider certificate 110c, an SRU-SP public key SRU-SP PK 106b, radio unit firmware 222, derived secret shared key 223, free memory 224 for SRU 109, and SRU firmware 109u. Data within memory 109r can be recorded in a file system similar to file system 229 for device memory 109g. OS 220 for SRU 109 can be an OS similar to the OS 101h for device 101,

but selected for the operating environment of SRU 109. Cryptographic algorithms 109q were described above in connection with Figure 1c. Device service provider certificate 110c can be used to verify a digital signature for data in a server B response 112b. SRU-SP public key SRU-SP PK 106b corresponds to SRU-SP private key SRU-SP SK 106a from Figure 1a. Radio unit
5 firmware 222 can comprise software, firmware, and operating libraries for radio unit 109b, including radio unit processor 109c.

Derived secret shared keys 223 can comprise secret shared keys derived between SRU 109 and SRU-SP 106 using an ECDH key exchange and SRU private key SRU SK 109p and SRU-SP public key SRU-SP PK 106b. Derived shared secret keys 223 can also comprise secret
10 shared keys derived with SRU-SP using an ephemeral private key derived by SRU 109 such as ephemeral private key 409a as depicted and described in Figure 4a and Figure 4b below. Free memory 224 for SRU 109 can comprise memory allocated for SRU 109 in protected memory 109r that is available for future use by SRU 109 during operations after a boot process. SRU firmware 109u is described above in Figure 1c and also below in Figure 2e. Other data could be
15 recorded in protected memory 109r as well, including an updated boot configuration 218, which could include boot parameters for the operation of RU 109 that are different than initially stored SRU boot configuration 218 in EEPROM 216.

Power control unit 109e can manage the power utilization by SRU 109. Since many applications of device 101 may operate on a battery 101n, current flow and energy consumption
20 by SRU 109 can be managed by power control unit 109e. Power control unit 109e can also include a voltage regulator in order to limit a maximum input voltage from pins 109x to components in SRU 109 (and can be connected to bus 202). Power control unit 109e can smooth or reduce voltage fluctuations from power received by device 101. In exemplary embodiments, power control unit 109e can either (i) disable power input into radio unit 109b or
25 (ii) force radio unit 109b to enter a sleep state when device 101 writes to memory 109a. The disabled or sleep state for radio unit 109b could comprise current utilization for radio unit 109b of less than an exemplary 5 micro amps. Power control unit 109e can also (i) change radio unit 109b from a booted but sleep state to (ii) an active mode or operating mode (e.g. ready to receive data and transmit data) with operating power for radio unit 109b after data is written by
30 device 101 into shared memory 101h, such as application data 112a from device 101 for server B 110a. In other words, power control unit 109e can keep radio unit 109b in a deep sleep state until SRU receives data from device 101 in shared memory 109h, where the data from device 101 in shared memory 109h is for radio unit 109b to transmit through wireless network 102.

Exemplary additional steps and function for a power control unit 109e are depicted and described in subsequent figures below, including Figure 6a and Figure 6b.

Further, power control unit 109e could disable writes or erases by device 101 into device memory 109g when radio unit 109b is transmitting, since those writes or erases in combination with radio unit 109b transmitting may exceed specified or preferred current consumption thresholds for storage radio unit 109. In exemplary embodiments, radio unit 109b can consume approximately 200 mA of current for several seconds when transmitting at full power of 23 dBm. Similarly, SRU 109a can consume approximately 50 -- 75 mA of current when erasing blocks of memory for device 101 in general memory 109g. Other possibilities exist as well, but those values are typical for radio units and flash memory units commercially available in 2018.

In addition, power control unit 109e could prevent writes or erases of data in device memory 109g when radio unit 109b is transmitting in order to reduce possible internal radio frequency noise. Power control unit 109e could control power input or supply voltage to radio unit 109b, such that input power to radio unit 109b must pass through power control unit 109e. Power control unit 109e could have electrical connections to radio unit 109b to signal sleep and wake signals, possibly through internal bus 212. For some exemplary embodiments, RU 109b can comprise a radio supporting 802.11 ah standards, which are also known as WiFi Halow. The sleep of RU 109b and/or SRU 109 can be controlled by a power control unit 109e, and power control unit 109e can receive a target wake time (TWT) from a wireless network 102. Power control unit 109e can power up or down the radio unit 109b and SRU 109 based on the TWT received. In exemplary embodiments, SRU 109 can write the TWT and a corresponding date/time value to device read memory 192, such that device 101 can read the TWT and plan or schedule to take sensor measurements based on the TWT received from the wireless network 102 by radio unit 109b.

Power control unit 109e could also utilize power detector 209. Power detector 209 could be connected to power control unit 109e via an internal bus 212. Power detector 209 can (i) detect when power or voltage is removed from pin 4 201g (or similar pins for voltage input with other busses) and (ii) send a shutdown signal to power control unit 109e. Power control unit 109e can receive the shutdown signal and conduct a clean shutdown of SRU 109 and radio unit 109b. For embodiments where RAM 213 comprises NVRAM, then a shutdown command from power control unit 109e after (x) power detector 209 determines that power from device 101 has been removed can then (y) instruct SRU 109 to store the last operating state in NVRAM (similar to a "hibernate" shutdown for a personal computer). Thus, SRU 109 can resume operation more quickly once power is restored to SRU 109 using the last operating state from NVRAM. Power

detector 209 and power control unit 109e operating together can remove power from RU 109b within fractions of a millisecond when RU 109b is transmitting and power is removed from SRU 109.

Power capacitor 208 could provide sufficient charge storage to allow SRU 109 to operate for at least a few milliseconds after power supply voltage is removed from pin 4 201g (or similar pins/connectors for voltage input with other busses). In this manner, power control unit 109e can complete outstanding write operations to memory 109f (e.g. memory 109h and memory 109g) or memory 109r, and avoid data being corrupted. In an exemplary embodiment, power capacitor could be a surface mount capacitor supporting approximately 20 - 200 micro farads at the low voltage needed by SRU 109, such as a maximum supporting voltage of 5 volts. SRU could consume an exemplary 25 - 50 mA during a device 101 write operation to memory 109f or radio transmission by RU 109b (such as with 0 – 15 dBm RF transmit power). Voltage supplied to SRU 109 can be an exemplary 3.3 volts.

As an exemplary calculation, if power was suddenly removed from pin 4 201g, then SRU could use charge stored in power capacitor 208 with an exemplary 100 microfarads and a voltage drop of 0.6 volts and a current draw of 33 mA to operate SRU 109 for an additional 2 milliseconds. SRU 109 using all of power detector 209, power capacitor 208, and power control unit 109e could reliably close memory write processes in less than 2 milliseconds, in order for SRU 109 to shut down without inducing errors in memory 109a or RAM 213 operating as NVRAM. However, under preferred operating conditions, SRU 109 receives both (i) a shutdown command from device 101 (equivalent to a “remove” or “dismount” command from device 101 for SRU 109) and (ii) at least 10s of milliseconds of power on pin 4 201g before device 101 terminates power input and supplied voltage to pin 4 201g. Power capacitor 208 could be utilized if power is suddenly removed from SRU 109 without a shutdown command from device 101. Note that power capacitor 208 could be used with power control unit 109e to smooth voltage fluctuations received through pins 109x, such that a more stable voltage can be supplied to components connected via internal bus 212.

In exemplary embodiments, power control unit 109e delays the completion of the startup of SRU 109 until after power capacitor 208 is sufficiently charged, such as being charged to a level of > 75% of capacity. In these exemplary embodiments, the delay of startup of SRU 109 from the application of power to SRU 109 from device 101 can normally be delayed to less than a few tens of milliseconds. The benefit of delaying startup of SRU 109 until power capacitor 208 is charged is that fluctuations in power from device 101 can be smoothed out, such as a transient power applied of a few milliseconds would not start SRU 109 in a state where SRU

109 cannot be cleanly shut down through the operation of power capacitor 208. Note that “power capacitor” herein contemplates a capacitor of sufficient size to power SRU 109 for at least a few milliseconds, which is different than a power capacitor for larger electrical circuits such as power supplies, which could use capacitors on the order of farads. Further, upon a power source with device 101 being removed, power control unit 109e could shut down radio 109b before shutting down processor 109c, while using stored electrical energy from power capacitor 208 in order to reduce current drain and allocate power resources to processor 109c cleanly shutting down processes when device 101 removes power from SRU 109.

Many of the logical steps for operation of SRU 109 and RU 109b can be performed in software and hardware by various combinations of processor 109d, processor 109c, firmware 109u or boot firmware 217 and other components in Figure 2a. When SRU 109 or RU 109b is described herein as performing various actions such reading a file, writing a file, transmitting or sending data, receiving data, encrypting a ciphertext, conducting a key exchange, etc., specifying herein that SRU 109 or RU 109b performs an action can refer to software, hardware, and/or firmware operating within SRU 109 illustrated in Figure 2a performing the action.

Figure 2b

Figure 2b is a graphical illustration of an exemplary system, where a device and a radio unit communicate through a shared nonvolatile memory in a storage radio unit, in accordance with exemplary embodiments. Device 101 and radio unit (RU) 109b can preferably communicate through an interconnected electrical interface and mutually accessible nonvolatile memory 109h in storage radio unit 109, in order to transfer data between the two elements. Exemplary embodiments for a system supporting communication through a shared nonvolatile memory 109h is illustrated in Figure 2b. Device 101 and RU 109b can both mutually access a shared memory 109h, which could comprise a selected portion of memory 109f, where memory 109f can comprise a selected portion of overall nonvolatile memory 109a as depicted and described in connection with Figures 1c and 2a above. A system 200b supporting a shared nonvolatile memory 109h can comprise a device 101, a shared memory 109h, an SRU external bus 202, and SRU internal bus 212, RU polling 232, device polling 233 and RU 109b. For some embodiments, mutually accessible nonvolatile memory 109h or shared nonvolatile memory 109h could operate in device 101 outside of a SRU 109, such as depicted in Figure 7 below.

A shared nonvolatile memory 109h can operate as a shared nonvolatile memory interface 230. A shared nonvolatile memory interface 230 can be referred to herein as a shared memory interface 230, where the shared memory interface 230 can operate in a nonvolatile memory

(such that data stored in the nonvolatile memory is recorded when power is removed from the memory cells recording the data). As contemplated herein, a shared memory interface 230 can be referred to as a “shared filesystem interface” 230’, for embodiments where shared nonvolatile memory 109h is formatted or structured with a filesystem for both device 101 and RU 109. In other words, a shared filesystem interface 230’ can be a subset or specific configuration for a shared memory interface 230, where a shared filesystem interface 230’ records a filesystem in a shared nonvolatile memory interface 230.

Note that the use of a shared filesystem interface 230’ as the structure for shared nonvolatile memory 109h is not required, and other structures are possible as well, such as simply addressing specific memory cells by both device 101 and RU 109b for the data depicted in Figure 2b. Some embodiments may use a shared filesystem interface 230’ within shared nonvolatile memory 109h in order to “reuse” widely adopted technology and existing, compatible technology of filesystem in device 101 and SRU 109 for addressing, reading, and writing to storage nonvolatile memory. Shared filesystem interface 230’ can also include a configuration process 231 and a device write and radio unit read memory 191 and device read and radio unit write memory 192. As contemplated herein, a “shared filesystem interface” 230’ may also be described or referred to as “shared filesystem” 230’.

The use of a shared filesystem 230’ for shared memory 109h can be desirable to establish communication between device 101 and RU 109b when (i) RU 109b resides on a storage radio unit 109 and (ii) SRU 109 operates with (i) established standards for removable media or (ii) external storage from CPU 101b, such as (a) using generic removable media drivers for device driver 101g and/or (b) using established filesystem and device drivers in device 101 for memory 109h and filesystem 230. The generic or pre-existing device driver 101g installed on a wide variety of devices 101 (or pre-included in various OS 101h for many different devices 101) may not include specific interfaces for communication with RU 109b operating inside a SRU 109, but communication between device 101 and RU 109b can be desirable in exemplary embodiments. For example, electrical pins 109x and device driver 101g could support a one-bit SD bus for removable media such as a micro SD card and resulting file or memory operations within the micro SD card (via SD card driver 101k), and SD card driver 101k included in device 101 may not require specific support to send data directly to RU 109. Data reporting libraries 101x could include the logic of using a standard device driver 101g such as an SD card driver 101k to write file or data to nonvolatile memory 109a, which SRU 109 could structure as a shared memory 109h, such that RU 109b could read the data written by data reporting libraries 101x.

The manufacturer of an SRU 109 may incur high commercial expenses in order to create device drivers 101g for the very wide variety of devices 101 and device OS 101h, as described in the needs of the art detailed above. However, the manufacturer of an SRU 109 could more quickly and for lower cost establish compatibility with a wide variety of devices 101 using an SD card driver 101k interface for SRU 109. Note that other common drivers besides SD card driver 101k could be utilized, such as a USB interface, PCIe interface, SPI interface, SATA interface, serial interface, etc.

Other possibilities exist as well, where a wide variety of manufactured devices 101 can (A) support standards-based storage media such as SD cards, but (B) lack pre-installed drivers 101g with protocols for communicating directly with RU 109b through a shared nonvolatile memory 109h. Storage radio unit 109 and RU 109b could subsequently utilize system 200b in order to communicate with device 101, where support for a device 101 to read and write to non-volatile memory 109a, memory 109f, and memory 109h could be widely supported by existing devices 101 and device OS 101h. Further, device 101 could use system 200b in order to communicate with radio unit 109b using a system 200b and a shared filesystem interface 230'.

A shared memory 109h with a shared filesystem interface 230' can be compatible with a generic or widely distributed media device driver 101g (such as SD card driver 101k). Use of a standard media device driver 101g can allow device 101 to (i) write data to memory 191 or files in storage radio unit 109, (ii) read data o from memory 192 or files in storage radio unit 109, and (iii) quickly and reliably mount or attach the shared memory 109h within an SRU 109. Using system 200b, RU 109b and device 101 can subsequently utilize this standard capability to communicate and share data, while SRU 109 also conducts or performs the function of nonvolatile memory storage for device 101.

In other words, for device 101 to send data to RU 109b in an exemplary embodiment, device driver 101g can write the data to memory 191 in shared memory 109h, where RU 109b can subsequently read data in the memory 191 in shared memory 109h using read libraries 217a within memory control libraries 217. Other, similar embodiments are possible as well, such as RU 109b using memory read and writes libraries in SRU firmware 109u (in which case read and write libraries in EEPROM 217 would be for boot functions). Or, SRU 109 using memory processor 109d and firmware 109u could conduct a read of memory 191 and send the data to RU 109b via internal bus 212.

RU 109b with processor 109c can conduct periodic polling 232 of files in memory 191 within shared memory 109h to determine that device 101 has written data for RU 109b to subsequently read (e.g. polling can detect if the timestamp for a file has been changed or

updated since a previous polling, or if a new file is available for reading by RU 109b). Further, in order for device 101 to read data from RU 109b, RU 109b could write the data to a file system in memory 192 in shared memory 109h using write libraries 217b within memory control libraries 217. Device 101 could subsequently read the data from memory 192 in shared memory
5 109h using device driver 101g. Device 101 can use periodic device polling 233 of memory 192 in shared memory 109h to determine that RU 109b has written data for device 101 to subsequently read.

As contemplated herein, RU polling 232 or device polling 233 could comprise a logical loop operating with processor 109c or processor 101b, respectively, where the loop determines
10 that a file or value recorded in memory has been updated or changed by the other node (where device 101 and RU 109b can comprise the two nodes). The loop timing for RU polling 232 or device polling 233 could operate at specified frequencies, such as an exemplary every 10 or 50 milliseconds, although other possibilities exist as well without departing from the scope of the present disclosure. In addition, other mechanisms exist for SRU 109 to use instead of a polling
15 232, such as the use of a trigger library 245 depicted and described in Figure 2e below. With a trigger library or function, data from device 101 passes through memory/interface processor 109d before being written to memory 109h, and processor 109d could include a library with a “trigger” such that processor 109d notifies processor 109c and/or RU 109b if a file is being written from device 101 to memory 191 for RU 109b to read.

In an exemplary embodiment, the memory/interface processor 109d could be coupled to
20 the power control unit 109e. The normal or default state for radio unit 109b could be an off or a sleep state and radio unit 109b could be in the sleep state. Data from device 101 to be stored in memory 109h (such as application data 112a) could trigger memory/interface processor 109d to signal power control unit 109e to wake radio unit 109b. Radio unit 109b could transition from
25 the sleep state to a wake or operational state and subsequently read the application data 112a and transmit to the wireless network 102. Or, a processor 109c could operate the trigger instead of a memory/interface processor, such that processor 109c wakes radio unit 109b upon receipt of application data 112a.

Before recording shared data in shared memory 109h, (i) SRU 109 or (ii) a configuration
30 unit 110d for device service provider 110 or (iii) a configuration unit for SRU-SP 106 could perform a shared memory filesystem formatting and configuration process 231. The process 231 could comprise the standard steps for partitioning, formatting nonvolatile storage media such as an SD card, where a file system is applied to the available memory. Process 231 could include formatting memory 109h to standards supported by device 101 such as exemplary file

systems of FAT16, FAT 32, exFAT, NTFS, ext3, ext4, exFAT, JFFS2, or UDF. Other file systems for memory 109f and memory 109h are possible as well without departing from the scope of the present disclosure. Process 231 could also format all of non-volatile memory 109a (which can include shared memory 109h). Additional details regarding a process 231 are provided in Figure 2f below. At the conclusion of a configuration process 231 on shared memory 109h, device 101 and RU 109b can mutually access a shared filesystem 230', where the shared filesystem 230' resides within storage radio unit 109 that also operates radio unit 109b.

In other words, device 101 or device service provider 110 can normally be responsible for selecting the format of a file system of memory 109g for general files 109g used by device 101. During a configuration process or step 231, the first portion of configuration 231 can comprise selecting, formatting, and writing a regular file system format for standard storage nonvolatile memory 109g or storage media, including removable media. In exemplary embodiments with the use of inserted media in device 101 such as, but not limited to, a SD card or micro SD card, the regular file system format can support existing SD cards that may not include a RU 109b in SRU 109. A second portion of configuration process 231 can comprise SRU 109 subsequently detecting and utilizing the selected file system from the first portion of a configuration process 231 over memory 109g for device 101.

In this manner RU 109b within a storage radio unit 109 can support the use of existing device drivers 101g in devices 101 (where device drivers 101g in device 101 are known to be compatible with the file system in memory 109g since memory 109g is structured to be read by device 101 using device drivers 101g). Communication between device 101 and RU 109b using a system 200b can utilize a device program 101i depicted and described in connection with Figure 1b for a device 101 (such as application data reporting libraries 101x) (i) to write data to memory 191 in shared memory 109h, (ii) read data from memory 192 in shared memory 109h, and (iii) conduct a periodic device polling 233. Additional details regarding a configuration process 231 are depicted and described in connection with Figure 2f below.

After a configuration 231 step, either device 101 or SRU 109 can load shared memory 109h with an initial set of values or files supporting the operation of RU 109b in SRU 109. In exemplary embodiments, the initial set of values or files for (x) shared memory 109h and a shared filesystem 230' could be included in (y) SRU firmware 109u. Additional details for firmware or libraries to support a shared filesystem 230' in shared memory 109h are depicted and described in connection with Figure 2e below. Device 101 could write data into device write and SRU read memory 191 in the format of either (i) multiple different files, or (ii) a single file with multiple values recorded in the file.

The file or files in memory 191 could include the depicted values of radio settings 191a, http(s) request/URL 191b, “mobile originated” (MO) SMS 191c, upload file 191d, cryptographic parameters 191e, signaling data 191f (to RU 109b), and control data 191g (for RU 109b). RU 109b could write data into RU write and device read memory 192 in the format of either (i) multiple different files, or (ii) a single file with multiple values recorded in the file. The file or files in memory 192 could include the depicted values of radio configuration 192a, http(s) response/file 192b, “mobile terminated” (MT) SMS 192c, download file 192d, cryptographic settings 192e, signaling data 192f (to device 101), and control data 192g (for device 101). Other possibilities exist as well for the values or data recorded in shared memory 109h without departing from the scope of the present disclosure, and the exemplary data above is illustrative as opposed to limiting. “Http(s)” shows that the data could be for transmission as either hypertext transfer protocol (http) or secured hypertext transfer protocol (https) such as using TLS or DTLS.

As an example of the data flow supported by a shared filesystem interface 230' in a system 200b, after configuration process 231, device 101 could write a MO SMS to 191c. RU 109b could conduct polling 232 in filesystem interface 230' within shared memory 109h to detect that a new MO SMS from device 101 is written to 191c. Power control unit 109e could power up components from a powered down or deep sleep state to a fully powered or operational state with RF front end 203, RF transceiver 204, eUICC 113, and baseband unit 206, such that RU 109b could connect with wireless network 102. RU 109b can subsequently read MO SMS from 191c and transmit the MO SMS to wireless network 102. After transmission of MO SMS to wireless network 102, power control unit 109e could power down for force components within RU 109b from the previous sentence into a powered off or deep sleep state if no inbound data for device 101 is received within an exemplary time such as 10 seconds (although other time values are possible as well, such as 3 or 15 seconds). The time settings for power down after transmission of the MO SMS by RU 109b could be specified in radio settings 191a. Other possibilities exist as well for both (i) the values in shared memory 109h and (ii) the steps of transferring the data between device 101 and RU 109b (and transfer of data to wireless network 102 and/or SRU-SP 106) without departing from the scope of the present disclosure.

Figure 2c

Figure 2c is a graphical illustration of a storage radio unit utilizing two clock inputs, in accordance with exemplary embodiments. SRU 109 can include a SRU memory/interface processor 109d in order to communicate with both (i) device 101 using an external bus 202 and

(ii) internal components of SRU 109 using an internal bus 212. As contemplated herein, and “external bus” 202 can comprise an external-facing bus 202 such as the electrical pins 109x, while external bus 202 may otherwise reside within SRU 109. Figure 2c depicts a solution to the needs in the art described above in the section “Description of Related Art”, where an SRU
5 109 may operate more efficiently, faster, and more reliably from using an internal clock source that is different than an external clock from device 101, where an external clock is input via a pin such as pin 5 for the micro SD card physical interface. The sources of timing for a memory/interface processor 109d depicted in Figure 2c can be utilized with other interfaces as well, such as, but not limited to, an SPI, PCIe, USB, serial interface, or enhanced SPI (eSPI).

10 In exemplary embodiments, a physical interface 202 between SRU 109 and device 101 can utilize a clock input 201e from the “master” or “host” (e.g. device 101). The flow of signals or electrical pulses between SRU 109 and device 101 can be timed based on clocking signals such as CMOS square waves on clock input 201e. These clock signals could be in exemplary ranges of 10 – 50 MHz for an SD bus interface operating on bus 202, where the speed is
15 determined by device 101 and derived from device clock 101m. The frequency of clock input 201e can comprise clock frequency 228, which can be input into processor 109d in order to operate interface driver 109y. In other words, interface driver 109y can be driven with the clock input 201e at a clock frequency of 228.

Although clock input 201e for SRU 109 can be required in order to transfer data between
20 SRU 109 and device 101 over bus 202, the frequency and stability of clock input 201e can be far from desirable for an optimal operation of SRU 109. For example, although clock input 201e could be suitable for communication via bus 202 and pins 109x, clock input 201e could have relatively high jitter, noise, phase variation, etc. Using a relatively “noisy” clock input 201e in SRU 109 to drive internal bus 212 and internal components in SRU 109 (such as RAM 213,
25 memory core interface 109b, etc.) could induce a higher level of errors for components in SRU 109 and connected via internal bus 212.

A typical clock input 201e from a low-cost device could have typical phase noise at the input frequency of -40 to -45 dBc/Hz. Note that significant “spikes” or “timing signal drops” much worse are possible, depending on the operation and power states of components in device
30 101, and these spikes of poor clock timing signals on clock input 201e could induce errors SRU 109. In contrast, internal RF clock 205 for SRU 109 could be a VTCXO or TCXO with a typical phase noise of -55 dBc/Hz at the output frequency. Note that SRU 109 controls RF clock 205 and thus can operate with a clock source with many fewer “spikes” or “drops”. Although the timing output of RF clock 205 may be set for RF transceiver 204 (such as an

exemplary 38.4 MHz with 4G LTE NB-IoT or CATM1), PLL 211 can scale clock 205 with low noise to a level such as an exemplary 192 MHz, where the output of PLL 211 can drive processor 109d as depicted in Figure 2c. The frequency of the output of PLL 211 as an input to processor 109b can comprise clock frequency 229, which can be an independent numerical value from clock frequency 228.

In exemplary embodiments and as depicted in Figure 2c, SRU memory/interface processor 109d can utilize clocking signal output from PLL 211 at a clock frequency 229 in order to drive (i) internal clock signals within processor 109d (excluding interface driver 109y) and (ii) internal bus 212 with internal bus driver 212a, and (iii) SRU 109 components attached to bus 212. In this manner, SRU 109 can control and/or reduce phase noise, “spikes” and “drops” in clocking signal timing from clock input 201e. Thus in exemplary embodiments, SRU processor 109d can utilize two clock inputs with two different frequencies, where the first clock input drives external facing bus 202 and the second clock input drives processor 109d and components on internal bus 212. In this manner, errors for communications and signals within processor 109d and electrical components in SRU 109 such as RAM 213 and nonvolatile memory 109a can be reduced.

Figure 2d

Figure 2d is a graphical illustration for using an interface between a device and a storage radio unit, in accordance with exemplary embodiments. Other interfaces besides interfaces for SD cards and micro SD cards can be used between device 101 and storage radio unit 109. System 200c in Figure 2d depicts the use of a serial peripheral interface (SPI), and other possibilities exist as well for the physical interface between device 101 and SRU 109 without departing from the scope of the present disclosure. In other words, the use of a filesystem interface 230' in shared memory 109h is not required for some exemplary embodiments, and the benefits of a system 100 in Figure 1, system 200a in Figure 2a, system 200b in Figure 2b, and system 400 in Figure 4, can be obtained without the use of a filesystem interface for a shared memory interface 230.

Device drivers 101g can include an SRU driver 234, where SRU driver controls the operation of SRU 109. SRU driver 234 could comprise a driver similar to a modem or module driver for a typical radio module for devices, with the addition of also controlling files in a nonvolatile memory of 109a of SRU 109. In other words, SRU 109 can include a radio and nonvolatile memory for device 101, and SRU driver 234 can comprise a driver using conventional technology for SRU 109 (e.g. a specific driver for the device OS 101h). When

using an SPI interface in system 200d, device 101 can also include an SPI master driver 235. SPI master driver 235 can include both an SPI memory library 235a and an SPI radio AT command library 235b.

SPI memory library 235a can support a first, memory interface 239a to memory core interface 109m operating in SRU 109. The memory interface 239a can pass through an SPI physical interface 236 for device 101, external pins 109x for storage radio unit 109, and SPI slave driver 237 in SRU 109 in order to communicate data with memory core interface 109m. Processor 109d can also be in the flow and control data sent and received over memory interface 239a between device 101 and memory core interface 109m. Device 101 can write and read data in nonvolatile memory 109a using memory interface 239a, where memory interface 239a uses an SPI interface. Other interfaces besides an SPI interface could be used as well, such as, but not limited to, a PCIe interface, serial interface, etc.

SPI radio AT command library 235b can support a second, AT interface 239b to radio unit 109b. The AT interface 239b can pass through an SPI physical interface 236 for device 101, external pins 109x for storage radio unit 109, and SPI slave driver 237 in SRU 109 in order to communicate data with AT command core 238. Processor 109d can also be in the flow and control data sent and received over memory interface 239b between device 101 and AT command core 238. AT command core 238 can drive and control radio unit 109b with conventional technology for an AT command set using AT command stack 109s, which is typical and standard for radio units, modules, and modems as of 2018. Device 101 can write AT commands to radio 109b via AT interface 239b and receive responses from radio 109b via AT interface 239b. In other words, AT interface 239b can operate as a traditional SPI interface for SRU 109 in some exemplary embodiments. Although an SPI interface is depicted in Figure 2d, other physical interfaces could be supported using the components and logic shown, such as an I2C, PCIe, PCI, UART, or serial interfaces, and other possibilities exist as well. An AT interface 239b and/or memory interface 239a could use a point-to-point protocol at a higher level than the physical layer in order to support communication between a device 101 and storage radio unit 109b.

Figure 2e

Figure 2e is a graphical illustration of storage radio unit firmware 109u, in accordance with exemplary embodiments. SRU firmware 109u can be recorded in nonvolatile memory 109r when SRU 109 is powered off, and an initial set of SRU firmware 109u could be loaded into SRU 109 by a manufacturer of SRU 109, distributor of SRU 109, SRU-SP 106, or possibly

device service provider 110. As described above in connection with Figures 1c and Figure 2a, SRU firmware 109u can be loaded by SRU 109 using boot firmware 217 and a boot configuration 218 recorded in EEPROM 216. SRU firmware 109u can include shared memory control libraries 240 for shared memory 109h, radio unit firmware 222 for radio unit 109b, an eUICC driver 249 for eUICC 113, an SD interface driver 109y for bus 202 and electrical pins 109x, SRU operating configuration 249b for SRU 109, and an operating system for SRU 109 comprising SRU OS 220. Although an SD interface driver 109y is depicted in Figure 2e, the interface driver could support other bus types communicating through electrical pins 109x.

Radio unit firmware 222 for radio unit 109b can include instructions and executable machine code and/or software libraries for processor 109c with radio unit 109b. Radio unit firmware 222 can include libraries for the operation of radio unit 109b, including an operating system for radio unit 109b. Radio unit firmware 222 can include firmware components to control and/or operate baseband unit 206, RF transceiver unit 204, RF front end 203, and interface with power control unit 109e and internal bus 212. Radio unit firmware can also include the logic and software/firmware for radio unit 109b to attach to wireless network 102 and transmit/receive data with wireless network 102 and also SRU-SP 106.

eUICC driver 249a can comprise instructions for radio unit 109b and processor 109c to communicate with eUICC 113. eUICC driver 249a can support an ES8 interface and data between eUICC 113 and an SM-DP (through radio unit 109b and wireless network 102), an ES5 interface and data between eUICC 113 and an SM-SR 107 (through radio unit 109b and wireless network 102), an ES10 interface and data between eUICC 113 and a local profile assistant operating in device 101 or SRU 109. In some exemplary embodiments, the communication of an ES10 interface for eUICC 113 can be with a server external to device 101, such as within SRU-SP 106. ES8 and ES5 interfaces and data are defined in GSM technical specification SGP.02. ES10 interface and data are specified in GSM technical specification SGP.22. eUICC driver 249a can also support communication between processor 109c in radio unit 109b and eUICC 113 such that eUICC 113 can operate with profiles 113c and 113d with network credentials and network parameters similar to a traditional SIM card in order for radio unit 109b to authenticate and connect with wireless network 102.

Interface driver 109y can comprise a driver for SRU 109 to communicate with device 101 via electrical pins 109x, and could comprise an SD bus driver for SRU 109 for SD card driver 101k in device 101. Other bus types and interfaces are possible as well for an interface driver 109y and driver 101k without departing from the scope of the present disclosure, and in exemplary embodiments driver 109y and driver 101k are of the same type and are compatible,

as well as the electrical signals through electrical pins 109x. Drivers 109y and 101k are also described above, such as with Figure 1b for device 101 with driver 101k and Figure 1c and Figure 2a for driver 109y.

SRU operating configuration 249b can comprise data, a file, or instructions for the operation of SRU 109. SRU operating configuration 249b can specify (i) the bus types for bus 202 and bus 212, (ii) values and settings for SRU 109 and radio unit 109b including timers, frequencies to utilize, memory allocations and settings for nonvolatile memory 109a, (iii) initial or power on settings for SRU 109, (iv) a list of supported capabilities for SRU 109, and (v) filesystem types, parameters, and partition information for data recorded in memory 109a, (vi) PLL 211 settings and clock settings, (vii) the settings or selection of a switch 203a to select an antenna for use by SRU 109. SRU operating configuration 249b can comprise a file such as an XML file, text file, and other possibilities exist as well without departing from the scope of the present disclosure.

In exemplary embodiments where SRU 109 includes a NVRAM for RAM 213 (e.g. in Figure 5a), after boot or an initial power up, SRU 109 records SRU operating configuration 249b in RAM 213 (shown in Figure 2a). RAM 213 can comprise an NVRAM, such that upon a subsequent power on, SRU 109 can read operating configuration 249b quickly from RAM 213 as NVRAM without first having to read and extract SRU operating configuration 249b from firmware 109u or flash nonvolatile memory 109r. SRU OS 220 can comprise an operating system suitable for SRU 109. Example operating systems 220 for SRU 109 include eCos, uC/OS, LiteOs, Contiki, OpenWRT, Raspbian, and other possibilities exist as well without departing from the scope of the present disclosure.

Shared memory control libraries 240 for shared memory 109h in SRU firmware 109u can include filesystem detection libraries 241, filesystem drivers 242, shared memory radio unit read library 243, device write directory filesystem structure 191x', device write directory trigger library 245, shared memory radio unit write library 246, and device write directory filesystem structure 192x'. Filesystem detection libraries 241 can read, detect, and determine supported filesystems in device general memory 109g such as filesystem 229 (shown in Figure 2a), where device general memory 109g can be configured, formatted, and/or partitioned by device 101, device service provider 110, or a user of device 101. Device general memory 109g can be configured by a configuration unit 110d or by device 101 directly. In exemplary embodiments, memory 109g can be configured in SRU 109 with a default file system such as a FAT 32 or an exFAT filesystem, although device 101 or a configuration unit 110d could select and write a different filesystem for memory 109g after distribution of SRU 109. Filesystem detection

libraries 241 can detect and determine a filesystem for memory 109g comprising filesystem 229 according to all available and supported filesystems in filesystem drivers 242. The use of a filesystem detection library 241 is also depicted and described in connection with step 255 in Figure 2f below.

5 Filesystem drivers 242 can comprise kernel modules for OS 220 in order for SRU 109 to access, read, and write files and data to memory 109f and memory 109r. Since device 101 can specify, format, and control a filesystem 229 in memory 109g within memory 109f, SRU 109 can preferably support a wide variety of filesystem drivers 242 in order for SRU 109 to read and write data and files for memory 109f, where the files and data can be subsequently be
10 compatibly accessed by device 101.

In other words, with conventional technology, a storage unit such an SD card with flash memory storage and a processor 109d and memory core interface 109m may operate with physical level or layer data and not operate with internal filesystem detection libraries 241 and filesystem drivers 242 in order for the SRU to write data (a) internally generated by the SRU
15 109 (or received by an NB-IoT network 104 or wireless network 102) and (b) for the device 101 to subsequently read. The source of data input for recording data in memory 109a is device 101 with conventional technology, while for exemplary embodiments the source of data input for recording data in memory 109a is internally generated by SRU 109.

Filesystem drivers 242 can support any standard filesystem used by a device 101 with
20 memory 109g and memory 109f, such as, but not limited to the filesystem types of FAT16, FAT32, exFAT, NTFS, Ext2, Ext3, Ext4, UDF, ISO-9660, HFX+, HFSX, Apple, UFS, XFS, JFS, YAFFS, JFFS2, UBIFS, and other possibilities exist as well, including subsequent, future, or related filesystems for these exemplary filesystems of a filesystem 229. In exemplary embodiments, SRU 109 can support a plurality of filesystem drivers 242 such as those depicted
25 in Figure 2f and select an individual filesystem driver 242 for OS 220 in SRU 109 to read and write files to memory 109h based on a detected filesystem for device 101 in memory 109g. In exemplary embodiments, not all depicted filesystem drivers 242 within SRU 109 are required, and a subset of the filesystems could be selected for SRU 109.

Read libraries 243 can provide machine executable code for processor 109d to read data
30 from device 101 using shared memory 109h. Read libraries 243 can be selected from filesystem drivers 242 to be compatible with filesystem 229 used by device 101 for memory 109g. The data read from shared memory 109h by read libraries 243 could be used with RU 109b or input into RU 109b. In this manner by using read libraries 243, memory 109h can provide the physical and logical interface for RU 109b to receive data from device 101. Read libraries 243

could specify memory addresses or file locations in a shared file system 230 for non-volatile memory 109h where device 101 can write data in order to be read by SRU 109 and RU 109b. In an exemplary embodiment, (i) device 101 could write a file with a name "http request" 191a (such a URL with parameters for server B 110a) to a specified location or file name in shared memory 109h, such as the depicted memory for device write and radio unit read memory 191 in Figure 2b above, and then (ii) read libraries 243 could instruct processor 109c or processor 109d to read the data in 191 and subsequently use data 191 (which could comprise the exemplary http request 191a) for transmission through radio unit 109b to SRU-SP 106 and/or server B 110a.

In exemplary embodiments, either (a) processor 109d could operate read libraries 242 and pass the data from memory 191 to processor 109c in radio unit 109b, or (b) processor 109c could operate read libraries 242 and read from memory 191 directly in order to pass the data 112a to wireless network 102. Other possibilities exist as well for an SRU 109 to read data input from device 101 into a shared filesystem 230' in storage radio unit 109 without departing from the scope of the present disclosure. Device write directory filesystem structure 191x' in device 101 (shown in Figure 1b) can specify the names of files and/or settings within a directory of shared filesystem interface 230' for device 101 to utilize in order for device 101 to send data to radio unit 109b. Device write directory filesystem structure 191x' can comprise the structure of a device write directory 191x with pre-agreed file names for data input from device 101 into SRU 109 for RU 109b. For example, the recoding the filesystem structure 191x' by device 101 is depicted and described for device 101 in Figure 1b. An exemplary device write directory 191x using a pre-agreed filesystem structure 191x' is depicted and described in connection with Figure 3a below.

Write libraries 246 can provide machine executable code for processor 109d to write data output from radio unit 109b to memory 109h and shared filesystem interface 230' in order for device 101 subsequently read for data input into device 101. Write libraries 246 can be selected from filesystem drivers 242 to be compatible with filesystem 229 used by device 101 for memory 109g. Note that filesystem 229 by device 101 can also be used as the filesystem for memory 101h. In this manner, write libraries 246 could provide the logical software or firmware interface for RU 109b to send data received from wireless network 102 to device 101, and other possibilities exist as well for the transfer of data from RU 109b to device 101 through a shared filesystem interface 230' in shared memory 109h without departing from the scope of the present disclosure.

Write libraries 246 could specify memory addresses or file locations in a file system 230 for non-volatile memory 109h where RU 109b using processor 109c can write data in order to

be subsequently read by device 101. In an exemplary embodiment, (i) RU 109b could write a file with a name "http response.htm" 192a (such as data received from server B 110a and/or SRU-SP 106 via wireless network 102) to a specified file location in memory 109h, such as the depicted memory 192 for device read and radio unit write operations, and then (ii) device 101
5 could subsequently read the data in 192 via bus 202 in order to obtain the server B response 112b data depicted in Figure 1a.

In exemplary embodiments, either (a) processor 109d could operate write libraries 246 and pass the data from processor 109c in radio unit 109b to memory 192, or (b) processor 109c could operate write libraries 246 and write to memory 192 in shared memory 109h directly in
10 order to pass the data 112b from wireless network 102 to device 101. Other possibilities exist as well for an SRU 109 to write data 112b output for device 101 into a shared filesystem 230' in storage radio unit 109 without departing from the scope of the present disclosure. Device read directory filesystem structure 192x' can specify the names of files within a directory of shared filesystem interface 230' for device 101 to utilize in order for device 101 to read data from radio
15 unit 109b. Device read directory filesystem structure 192x' can comprise settings for a directory with pre-agreed file names for data input from RU 109b (and/or SRU 109) into shared memory 109h or shared filesystem 230' for device 101 to subsequent read as data 112b input into device 101.

Note that device 101 can also store and operate with a device read filesystem structure
20 192x' in device program 101i including data reporting libraries 101x. An exemplary device write directory filesystem structure 192x is depicted and described in connection with Figure 3b below. In exemplary embodiments the separate read and write functions in SRU firmware 109u in Figure 2e and separate read and write memory in shared memory 109h could be combined into a single read/write function in SRU firmware 109u and a single read/write memory or set of
25 files in shared memory 109h.

In an exemplary embodiment described above for Figure 2e, SRU 109, using read libraries 243 and write libraries 246, can (i) read data from device 101 using shared memory 109h and (ii) write data to memory 109h, where device 101 can subsequently read data from memory 109h. In another embodiment, write libraries 246 could provide instructions for SRU
30 to write data 112b directly to RAM 101e or processor 101b, but in this case a closely coupled software and firmware interface between SRU 109 and device 101 through electrical pins 109a and/or bus 202 would be required. For example an SPI bus mode, a one-bit SD bus mode, or a four-bit SD bus mode for the interface through electrical pins 109a would not normally be able to support SRU 109 writing data directly to RAM 101e or processor 101b without instruction

from device 101 (e.g. SRU 109 with conventional technology is a “slave” or “client” and device 101 is a “master” or “host”), so alternative bus technology connecting storage radio unit 109 and device 101 could be utilized for SRU 109 to write data directly to RAM 101e or processor 101b (without instruction from device 101).

5 Trigger library 245 can comprise executable machine code for SRU 109 and processor 109d to automatically detect when device 101 writes data to shared memory 109h and/or shared filesystem 230' with device write memory 191. Processor 109d using a trigger library 245 could provide an alternative to RU 109b polling 232 depicted in Figure 2b above, and in this manner RU polling 232 could be bypassed and processor 109d could signal RU 109b when data is
10 available in memory 191 for RU 109b to read, process, and/or transmit to wireless network 102. In exemplary embodiments, trigger library 245 is coupled with power control unit 109e (from Figure 1c and Figure 2a above), such that power control unit 109e (a) instructs RU 109b to enter a sleep mode (without requiring polling 232) when no new or “untransmitted” data 112a is available from device 101 in memory 109h and shared filesystem 230', and (b) instructs RU
15 109b to wake or enter an active mode when new data 112a has been written by device 101 to memory 109h as detected and processed by a trigger library 245. After a confirmation or response data 112b is received by SRU 109 and written to shared memory 192, power control unit 109e could instruct radio unit 109b to enter a sleep or deep sleep mode or state, such as consuming less than an exemplary 5 microamps.

20 Note that the use of a trigger library 245 and a power control unit 109e in conjunction can help minimize the time radio unit 109b remains in an active state, where data 112b can be recorded in nonvolatile memory 192 after receipt by RU 109b from wireless network 102, and RU 109b can consequently enter a sleep mode without having to wait for device 101 or device CPU 101b to read the data 112b. Device 101 can read the server/network response data 112b
25 from memory 192 in memory 109h at a time that is convenient and/or optimized energy utilization for device 101 and CPU 101b. In an exemplary embodiment, device 101 could enter a sleep state while RU 109b is sending data 112a and receiving data 112b (as depicted and described in connection with Figure 4a below), and SRU 109 would prefer to not wait until device 101 wakes in order to pass server B response data 112b to device 101.

30 In other words, through the use of a nonvolatile memory 192 to store or record (a) server/network response data 112b in response to (a) device 101 transmitted data 112a, the amount of time that CPU 101b is active and RU 109b is active can be reduced compared to conventional technology, since the communication between CPU 101b and RU 109b can take place through a nonvolatile shared memory interface. Likewise, for device transmit data 112a

(e.g. application data 112a), CPU 109b in device 101 could write the data in memory 191 for memory 109h while RU 109b sleeps. Then, CPU 109b can essentially immediately enter a sleep state and remain in a sleep state until (i) RU 109b writes server B response data 112b in memory 192 and SRU 109 can (ii.a) send a signal via pins 109x for CPU 109b to wake and read data 112b from memory 192 and (ii.b) sleep RU 109b essentially immediately after data 112b is written to memory 192 in shared memory 109hs.

Figure 2f

Figure 2f is a flow chart illustrating exemplary steps for configuring a storage radio unit to communicate with a device using a shared nonvolatile memory interface, in accordance with exemplary embodiments. In order for storage radio unit (SRU) 109 to communicate with device 101 through a shared memory 109h and a shared nonvolatile memory interface 230 (which can comprise a shared filesystem interface) depicted described above, including in Figure 1c and Figure 2a, an SRU 109 can conduct a configuration process with steps depicted in Figure 2f.

The processes and operations, described below with respect to all of the logic flow diagrams may include the manipulation of signals by a processor and the maintenance of these signals within data structures resident in one or more memory storage devices. For the purposes of this discussion, a process can be generally conceived to be a sequence of computer-executed steps leading to a desired result.

These steps usually require physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is convention for those skilled in the art to refer to representations of these signals as bits, bytes, words, information, elements, symbols, characters, numbers, points, data, entries, objects, images, files, or the like. It should be kept in mind, however, that these and similar terms are associated with appropriate physical quantities for computer operations, and that these terms are merely conventional labels applied to physical quantities that exist within and during operation of the computer.

It should also be understood that manipulations within the computer are often referred to in terms such as listing, creating, adding, calculating, comparing, moving, receiving, determining, configuring, identifying, populating, loading, performing, executing, storing etc. that are often associated with manual operations performed by a human operator. The operations described herein can be machine operations performed in conjunction with various input provided by a human operator or user that interacts with the computer.

In addition, it should be understood that the programs, processes, methods, etc. described herein are not related or limited to any particular computer or apparatus. Rather, various types of general purpose machines may be used with the following process in accordance with the teachings described herein.

5 The present disclosure may comprise a computer program or hardware or a combination thereof which embodies the functions described herein and illustrated in the appended flow charts. However, it should be apparent that there could be many different ways of implementing the disclosure in computer programming or hardware design, and the disclosure should not be construed as limited to any one set of computer program instructions.

10 Further, a skilled programmer would be able to write such a computer program or identify the appropriate hardware circuits to implement the disclosed disclosure without difficulty based on the flow charts and associated description in the application text, for example. Therefore, disclosure of a particular set of program code instructions or detailed hardware devices is not considered necessary for an adequate understanding of how to make and
15 use the disclosure. The inventive functionality of the claimed computer implemented processes will be explained in more detail in the following description in conjunction with the remaining Figures illustrating other process flows.

Further, certain steps in the processes or process flow described in all of the logic flow diagrams below must naturally precede others for the present disclosure to function as described.
20 However, the present disclosure is not limited to the order of the steps described if such order or sequence does not alter the functionality of the present disclosure. That is, it is recognized that some steps may be performed before, after, or in parallel other steps without departing from the scope and spirit of the present disclosure.

The processes, operations, and steps performed by the hardware and software described
25 in this document usually include the manipulation of signals by a CPU or remote server and the maintenance of these signals within data structures resident in one or more of the local or remote memory storage devices. Such data structures impose a physical organization upon the collection of data stored within a memory storage device and represent specific electrical or magnetic elements. These symbolic representations are the means used by those skilled in the art
30 of computer programming and computer construction to most effectively convey teachings and discoveries to others skilled in the art.

At step 251, SRU 109 can be manufactured w/ radio unit 109b, SRU ID 109i, nonvolatile memory 109a, SRU only accessible memory 109r, and firmware 217, 218, and 109u. Other electrical components for an SRU 109 depicted in Figure 2a could be included in a

manufacturing step 251 as well. SRU firmware 109u as depicted in Figure 2e above can include an operating system 220 and filesystem drivers 242 for SRU 109, as well as the other data depicted for SRU firmware 109u. An SRU-SP 106 could then receive the manufactured SRU 109 units from a step 251 and conduct a step 252 to write configuration data to SRU 109. Other entities such as a device service provider 110 or distributor of SRU 109 could conduct a step 252 as well. At step 252, SRU 109 records SRU private key SRU SK 109p, eUICC bootstrap profile 113c, and SRU-SP certificate with an SRU-SP public key 106b. For a step 252, SRU 109 could derive SRU SK 109p and SRU PK 109w, using the same steps and data to derive ephemeral PKI keys in a step 409 below in Figure 4 (but with different PKI keys output). SRU PK 109w could be written to memory 192 after a set of configuration step 231 format or structure the shared memory 192. SRU firmware 109u could be recorded in a step 252 instead of a step 251, and other possibilities for the sequence of steps depicted in Figure 2f are possible as well without departing from the scope of the present disclosure. A configuration unit similar to configuration unit 110d in Figure 1a could conduct step 252 with SRU 109.

At step 253, SRU 109 can be distributed and inserted into a device 101. SRU 109 could be inserted into device 101 during manufacturing of device 101, or after device 101 is distributed to users or organizations owning device 101, where the user or entity owning device 101 inserts SRU 109. For example, with embodiments where SRU 109 comprises a micro SD card form factor such as within a housing 227, SRU 109 could be inserted into device 101 in a step 253. SRU 109 could comprise an integrated circuit or wafer level chip scale package for inclusion in device 101, and SRU 109 could be soldered onto a circuit board or inserted into a larger wafer level package in step 253. SRU 109 could comprise a module or components with a larger, integrated circuit comprising a system on a chip 801 from Figure 8, where a step 253 can be including the components for an SRU 109 on the system on a chip 801. As depicted in Figure 2f, the collection of steps 251 through 253 can comprise a step 260.

At step 254 a filesystem 229 can be written to memory 109g in nonvolatile memory 109a of storage unit 109. The exemplary filesystem 229 could comprise any of the exemplary file systems depicted and described in connection with Figure 2e above. A configuration unit 110a for device service provider 110 could write the filesystem 229 to memory 109g. Or, device 101 could perform the steps of a filesystem 229 configuration on SRU 109, equivalent to a personal computer formatting a USB memory stick or SD card with a file system. Other possibilities for conducting a step 254 include a manufacturer of SRU 109 or SRU-SP 106 (i) creating a default filesystem for SRU 109, which could be an exFAT or FAT32 file system, and (ii) writing the default filesystem to memory 109g. For example, SRU-SP 106 could create filesystem 229

when data is written to SRU 109 in a step 252 above. General files for device 101 could also be written in a step 254, such as writing a device OS 101h within the filesystem 229. Other data, including device reporting libraries 101x and device drivers 101g can be recorded in the filesystem 229 in a step 254. Note that a step 254 can be conducted before a step 253, such that
5 memory 101g with filesystem 229 includes data for device 101 to operate before SRU 109 is inserted into device 101.

At step 255, SRU 109 can receive power from device 101 and boot with boot firmware 217 and boot configuration 218. SRU 109 can load firmware 109u in order to begin operating. SRU 109 can conduct a step 255 with filesystem detection libraries 241 and filesystem drivers
10 242 in order to determine and read from the filesystem 229 previously written to device general memory 109g in step 254. In other words, processor 109d can (i) read the locally stored files in memory 109a and memory 109g using filesystem detection libraries 241 in order to (ii) determine the type of filesystem 229. The determined or detected type of filesystem 229 from a step 255 could comprise any of the listed exemplary file systems from Figure 2d above. Note
15 that SRU 109 can conduct a step 255 without requiring any input from device 101 during a step 255, other than device 101 providing an electrical power source.

At step 256, SRU 109 can select a file system driver 242 based on the detected and determined file system 229 from a step 255. At step 257, SRU 109 can write SRU 109 device write directory 191x to shared memory 109h using the selected file system driver 242. A step
20 257 can also include SRU 109 conducting a filesystem formatting process, in order to write the filesystem 229 to memory 191 in shared memory 109h. Or, in a step 257, SRU 109 can write files for memory 191 in an existing filesystem detected for filesystem 229. In other words, SRU 109 can add a directory 191x to an existing filesystem 229, and the directory 191x can comprise device write memory 191. Thus, a directory 191x could comprise a directory added to an
25 existing filesystem 229, where the directory 191x could comprise a portion of the shared nonvolatile memory 109h.

SRU 109 can use (i) memory write libraries 217c from boot firmware and (ii) memory core interface 109m in order to change NAND and/or NOR flash memory gates, or memory blocks in order to write binary data to memory 109h, where the resulting binary data written can
30 comprise a device write directory 191x for the file system 229. Device write directory 191x at step 257 can also include base data such as the exemplary files and data depicted in Figure 3a below. The file permissions 301a for a device write directory (in Fig. 3a below) can be set at the same time as a step 257. In addition, user a 302a and user b 302b can be specified (shown in Figure 3a below). Further, a step 257 can comprise SRU 109 specifying that user a 302a for file

permissions 301a in device write directory 191x can be the same as user a 302a for filesystem 229 in general memory 109g for device 101. In this manner, device 101 can determine that OS 101h also has write access to directory 191x (since OS 101h or the equivalent user can be the user a 302a in a filesystem 229 for memory 109g).

5 At step 258, SRU 109 can write SRU write directory 192x to shared memory 109h using the selected file system driver 242. Or, in a step 258, SRU 109 can write files for memory 192 in an existing filesystem detected for filesystem 229. In other words, SRU 109 can add a directory 192x to an existing filesystem 229, and the directory 192x can comprise device write memory 192. SRU 109 can use (i) memory write libraries 217c from boot firmware and (ii)
10 memory core interface 109m in order to change NAND and/or NOR flash memory gates, or memory blocks in order to write binary data to memory 109h, where the resulting binary data written can comprise a RU write directory 192x for the file system 229. RU write directory 192x (or equivalently device read directory 192x) at step 258 can also include base data such as the exemplary files and data depicted in Figure 3b below. The file permissions 301a and 301b
15 for a RU write directory (in Fig. 3a below) can be set at the same time as a step 258. In addition, user a 302a and user b 302b can be specified.

Further, a step 258 can comprise SRU 109 specifying that user a 302a for file permissions 301a in RU write directory 192x can be the same as user a 302a for filesystem 229 in general memory 109g for device 101. In this manner, device 101 can determine that OS 101h
20 also has read access to directory 192x (since OS 101h can be the user a 302a in a filesystem 229 for memory 109g). As depicted in Figure 2f, the collection of steps 255 through 258 can comprise a configuration step 231, and a configuration step 231 is also depicted and described in connection with Figure 2b above. Configuration step 231 can also be referred to as a configuration process 231.

25 At step 259, device 101 driver 101k can record or store device write files in directory 191x, such as data from reporting libraries 101x which could also comprise application data 112a from Figure 1a above. As noted above in Figure 2e, CPU 101b can sleep after writing data to directory 191x, since the data is stored in a nonvolatile flash memory and RU 109b can read the data at a later time after CPU 101b enters a sleep state. Note that power to SRU 109 via
30 electrical pins 109x can continue after CPU 101b enters a sleep state. A power control unit 109e can detect new data in directory 191x, such as from a trigger in a trigger library 245, and power up and change RU 109b from a sleep state to an active state. After either (i) a RU polling 232 detection of new data in directory 191x or (ii) a trigger library 245 detection or recording of new data in directory 191x, RU 109b can subsequently read the data from directory 191x and process

the data. If the data written to directory 191x comprises application data 112a, then RU 109b can transmit the data to a wireless network 102.

Figure 3a

Figure 3a is a graphical illustration of a shared filesystem interface for a radio storage unit and a device, using a device write directory, in accordance with exemplary embodiments. SRU 109 can operate and record a device write and RU read memory 191, where memory 191 can reside within shared nonvolatile memory 109h (as depicted in Figure 2a). Shared nonvolatile memory 109h could be formatted with a shared filesystem 230' using the configuration steps 231, such that device 101 could write to shared filesystem 230' and RU 109b and SRU 109 could read from shared filesystem 230'. In addition, memory 191 can be referred to as "radio unit read and device write memory" 191 or simply memory 191. Shared filesystem memory 230 device write directory 191x is depicted as viewable and readable from device operating system 101h with a file path "../device_write/.." although other file paths are possible as well.

Note that shared filesystem 230' could also be presented to device operating system 101h after configuration process 231 as a separate drive or mounted filesystem for device 101. Device write directory 191x could also be referred to as RU read directory 191x. Or, directory 191x could appear in operating system 101h as a new directory 191x in existing filesystem 229 for device 101 after a configuration step 231. In addition, any files recorded in shared filesystem 230' may preferably be recorded in a nonvolatile state, such that the files remain available to device 101 and SRU 109 and RU 109b when (i) power is removed from device 101 and/or SRU 109 and RU 109b, and (ii) power is subsequently provided to device 101 and/or SRU 109 and RU 109b.

Device write directory 191x could include a plurality of different files 191y over the lifetime of operation of SRU 109 and device 101. Although a plurality of different files 191y are depicted in Figure 3a, not all files may be present in directory 191x within shared filesystem 230' at the same time and the different files depicted within device write directory 191x in Figure 3a could be written by device 101 and read by RU 109b at different points in time. In some exemplary embodiments, such as those depicted in Figure 3a, directory 191x could be initialized after a configuration process 231 with an empty set of files, where SRU 109 and RU 109b can wait for device 101 to send files, data, or commands or data for RU 109b within SRU 109 in the form of at least one file 191y.

Exemplary files for a device write directory 191x within device write and RU read memory 191 can comprise files 191y. Exemplary files 191y can comprise files for radio_parameters_input 191a, http(s)_request 191b, MO_sms 191c, FTP_file_upload 191d, command_script 191g, power_control_settings 191f, device_identity 101a, cryptographic_settings 191e, and date-time-set 191h. Other files or data for a device write directory 191x are possible as well without departing from the scope of the present disclosure. The files 191y can comprise text, numeric, or binary data according to encoding rules such as using URL encoding with http(s)_request, and UTF-8 for MO_sms 191c, and other possibilities exist as well. Device reporting libraries 101x in device 101 and radio unit read library 243 could agree or be configured to use a mutually compatible encoding rules for each of the files 191y and/or file types in directory 191x.

Radio_parameters_input 191a in files 191y can comprise data in a file format for radio settings 191a depicted in Figure 2b. Values could specify frequencies for RU 109b to utilize, RF networking technology to utilize, such as NB-IoT 104 or LTE Cat M1 or other settings as well. Radio_parameters_input 191a could specify a server B 110a name or address to utilize for the transmission of application data 112a through radio unit 109b, wireless network 102, and SRU-SP 106. Radio_parameters_input 191a could also include values for timers and counters for radio unit 109b to utilize, such as a retry timers and counters for the transmission of application data 112a. For embodiments where SRU 109 includes more than one eUICC 113, radio_parameters_input 191a can specify a preferred or primary eUICC 113 for RU 109b to utilize, where each different eUICC 113 could record and operate with a different operational profile 113d. In this manner, device 101 could select a set of network access credentials and network parameters for RU 109b.

Http(s)_request 191b, MO_sms 191c, and FTP_file_upload 191d could comprise the file format for stored data in a shared filesystem interface 230' equivalent to the values and data depicted and described for device write and radio unit read memory 191 in Figure 2b. Device 101 could write other application data 112a to device write directory and device write files 191y as well, including an outbound email message, a Message Queuing Telemetry Transport (MQTT) message, a Session Initiation Protocol (SIP) message, and other possibilities exist as well without departing from the scope of the present disclosure. Http(s)_request 191b, MO_sms 191c, and FTP_file_upload 191d and the other listed exemplary messages from the previous sentence individually or together could comprise exemplary application data 112a from Figure 1a, as depicted in Figure 3a.

Other possibilities exist as well for application data 112a in a file 191y in directory 191x for shared filesystem interface 230' without departing from the scope of the present disclosure. Application data 112a could also be written by device 101 in an encrypted form and/or comprise data for RU 109b to send to server B 110a as TCP or UDP datagrams. Files in application data 112a could include destination TCP or UDP port numbers to utilize for RU 109b to transmit the data to wireless network 102. Although single values of the files 191y for application data 112a are depicted in Figure 3a, a directory 191x could record multiple different values of the depicted files (such as multiple http(s)_request 191b files 191y for RU 109b to transmit or multiple different MO_sms 191c for RU 109b to transmit, etc.)

Command_script 191g can comprise a file with a set of sequential or programmatic commands for RU 109b to conduct, and could be equivalent or similar to an AT command script. Command script 191g could instruct RU 109b to periodically attach to wireless network 102 and query server B 110a (possibly through SRU-SP 106) in order to check for or obtain new server B data 112b for device 101. Command_script 191g could instruct RU 109b for actions for RU 109b to take when no connectivity to a primary or preferred wireless network 102 is available, including the steps to search for backup wireless networks 102. Command script 191g could also instruct RU 109 to periodically attach to wireless network 102 and query a different server such as server A 108.

In some exemplary embodiments, command_script 191g can comprise a traditional AT command script for RU 109b to execute. AT command script could use instructions from standards such as 3GPP TS 27.007 or TS 27.005, as well as related or subsequent versions of these standards. In some exemplary embodiments, the command_script 191g could include application data 112a within the script 191g, such that a single file or sequence of data can be passed to RU 109b, where the single file or sequence of data includes commands to attach to wireless network 102 and transmit the application data 112a for the command script. Or, the command_script 191g could refer to application data 112a in the form of a file 191y for RU 109b to transmit.

In another exemplary embodiment, processor 109d could read command_script 191g recorded by device 101 and write the command_script 191g to RU 109b using an internal bus 212, where bus 212 could operate with a traditional UART, serial, or SPI interface for RU 109b. In this manner, (i) commercially available radio units or modules with AT command interfaces could be utilized with an RU 109b while (ii) obtaining the benefits of a shared memory interface 230 to transfer the data from CPU 101b in device 101 and RU 109b. Processor 109d could also read application data 112a from a file 191y and pass or write the data to RU 109b. Or, RU 109b

could “fetch” or read the application data 112a in the form of a file 191y based on a file path passed along in the command_script 191g. A benefit of using command_script 191g from nonvolatile memory 109h is that CPU 101b can be in a sleep state when RU 109b reads the data and performs the operations specified by the command_script 191g. Other possibilities and benefits exist as well for a directory 191x in a shared filesystem interface 230’ to record a command_script 191g for RU 109b without departing from the scope of the present disclosure.

In some exemplary embodiments, command_script 191g could be recorded in a format other than a file 191y within memory 191, such as a sequence of binary data with a start and end memory address within memory 109g, and processor 109d could use libraries 243 and 246 to track the start and end memory addresses. As depicted and described in connection with Figure 2b, data in memory 109h such as device write memory 191 and RU write memory 192 can be recorded without using a file system. Other data depicted in Figure 3a could be recorded in memory 191 without using or requiring a traditional file system 229 for memory 191 and memory 192.

Power_control_settings 191f as a file 191y can include settings or values for use with a power control unit 109e by SRU 109 or RU 109b. Power_control_settings 191f could specify the time for RU 109b to remain in an active state or a connected state with wireless network 102 after transmitting application data 112a in order to receive a server B response 112b, such as an exemplary 5 seconds or an exemplary 10 seconds. Power_control_settings 191f could specify the use and values for a power saving mode (PSM) used by radio unit 109b, as well as a frequency or time for radio unit 109b to utilize a discontinuous receive (DRX) mode associated with wireless network 102. Power_control_settings 191g could specify a maximum power utilization by both SRU 109 in total and RU 109b. Power_control_settings 191f could specify the maximum RF power output for RU 109b to utilize in dBm or watts. In another exemplary embodiment, power_control_settings 191f can provide settings or commands for the operation of power control unit 109e.

Device_identity 101a written by a device 101 as a file 191y can comprise the identity of the device 101, which could be sent or transmitted by SRU 109 and RU 109b to SRU-SP 106 and/or server B 110a. Although not depicted in Figure 3a, a file 191y with a digital signature for device 101 could be written to device write directory 191x as well as a file 191y. The digital signature could be over a file or data in application data 112a for transmission to SRU-SP 106 and/or server B 110a through wireless network 102, and SRU-SP 106 or server B 110a could verify the digital signature for the device using a certificate for device 101 recorded in SRU-SP 106 or server B 110a, where the digital signature could be used by the receiving party of

application data 112a along with receipt of device_identity 101a to select the certificate for device 101 and verify the digital signature created by device 101. In this manner, SRU-SP 106 or server B 110a could verify that the application data 112a received by SRU-SP 106 or server B 110a was authentically originated by device 101.

5 Cryptographic_settings 191e can comprise values, parameters, public keys, and symmetric ciphering keys for RU 109b to utilize. Cryptographic_settings 191e can also be referred to as cryptographic parameters 191e. In an exemplary embodiment, cryptographic_settings 191e could include a public key or certificate for device 101, where device 101 continues to record and operate with the corresponding private key.
10 Cryptographic_settings 191e could specify parameters or values for SRU 109 and/or RU 109b to use with cryptographic algorithms 109q as depicted and described in connection with Figure 1c above. Cryptographic_settings 191e could select algorithms and/or key lengths to use with an ECDH key exchange, an ECC named curve such as, but not limited to, P-256, secp256k1, P-384, curve25519, and other possibilities exist as well, including algorithms supporting post-
15 quantum cryptography.

Cryptographic_settings 191e could specify which of a plurality of different possible public keys for device 101 recorded in memory 109h that SRU 109 or RU 109b should utilize with cryptographic_algorithms 109q. Cryptographic_settings 191e could also specify which of a plurality of possible secure hash algorithms SRU 109 or RU 109b should utilize in
20 communications over wireless network 102, such as, but not limited to SHA-256, SHA-384, RIPE-160, SHA-3, etc. Cryptographic_settings 191e as a file 191y could also select the use of a symmetric ciphering algorithm for SRU 109 or RU 109b to utilize with communications over wireless network 102, such as, but not limited to, AES-128, 3DES, AES-192, AES-256, blowfish, as well as specifying a cipher mode such as AES-CBC, AES-CTR, AES-SIV, etc.

25 A date-time-set file 191h could specify the current date and time for SRU 109. In exemplary embodiments, device 101 can record a clock 101m which continues to keep time when device 101 is powered down or off, such as a system time for a personal computer that continues to increment even when the personal computer is powered off (although a battery such as battery 101n could continue to provide a lower level of power in order for device 101 to
30 continue to keep track of time). Since SRU 109 may also be powered down when device 101 is powered down or off, SRU 109 may need an updated value for data and time when SRU 109 is powered up. The value for date and time for SRU 109 and/or RU 109b to utilize can be set by a file 191y with the current date and time. SRU 109 and RU 109a can utilize the date and time from a date-time-set 191h (along with an internal clock for SRU 109 from clock 207 or clock

205) to create timestamp 304a with files 191y recorded in directory 191x. In another exemplary embodiment, RU 109b can obtain the current date and time from wireless network 102, and utilize that date and time with RF clock 205 in order to continuously monitor and track time when SRU 109 is powered. In exemplary embodiments, (i) device 101 writes a value for date and time 191h with application data 112a before CPU 101b sleeps, then (ii) SRU 109 reads the date and time 191h and keeps time using an internal clock such as 205, and then (iii) SRU 109 uses the subsequent date and time calculated from the internal clock to write the date and time values associated with files written by SRU 109 in directory 192x (shown below in Figure 3b).

File system interface 230 in shared memory 109h can also include file parameters or attributes. File parameters may be specific to a selected file system driver 242 and selected file system 229 for directory 191x in Figure 3a and 192x below in Figure 3b, and the depicted file parameters in Figure 3a are exemplary. File parameters could include permissions 301a, which include values to specify permissions for users 302a and 302b with the file, such as read, write, execute, etc. Different permissions in permissions 301a could be associated with different users 302a and 302b. Users 302a and 302b could comprise users for directory 191x, such as a device OS and radio unit OS. Depicted users for users 302a and 302b are exemplary for Figure 3a. Note that the exemplary files 191y in directory 191x depict two users 302a and 302b comprising a first user of device 101 (identified as the OS 101h) and a second user of SRU 109 (identified as the OS for RU 109b). Other possibilities exist as well without departing from the scope of the present disclosure. The user 302a could be identified and selected from a user for files in memory 109g and filesystem 229.

File parameters for file system interface 230 can include a file size 303a, which can comprise the number of bytes for the recorded file 191y. Timestamp 304a can comprise the date and time when the file 191y was written in directory 191x. Timestamp 304a can be written by OS 101h. Although only a year is depicted in Figure 3a for some files, a time value in hours, minutes, and seconds could also be recorded. File parameters or attributes can also include the file name 305a. The file names 305a and file structure (e.g. encoding, XML fields, JSON fields, etc.) within files 191y and directory 191x could be pre-agreed via a specification mutually shared between device 101 and SRU 109 before device 101 begins operating with SRU 109. In addition, although device write directory 191x and RU write directory 192x are depicted in this Figure 3a and Figure 3b below as separate directories within shared filesystem 230', the two directories could be combined, and device write files 191y and RU write files 192y (below in Figure 3b) could be combined into the same, single directory within shared filesystem 230'.