US 20150121066A1

(19) United States

(12) Patent Application Publication

Nix

(10) Pub. No.: US 2015/0121066 A1

(43) Pub. Date: Apr. 30, 2015

(54) SET OF SERVERS FOR "MACHINE-TO-MACHINE" COMMUNICATIONS USING PUBLIC KEY INFRASTRUCTURE

(71) Applicant: **John A. Nix**, Evanston, IL (US)

(72) Inventor: **John A. Nix**, Evanston, IL (US)

(21) Appl. No.: **14/064,618**

(22) Filed: **Oct. 28, 2013**

**Publication Classification**

(51) **Int. Cl.**

*H04L 9/32* (2006.01)

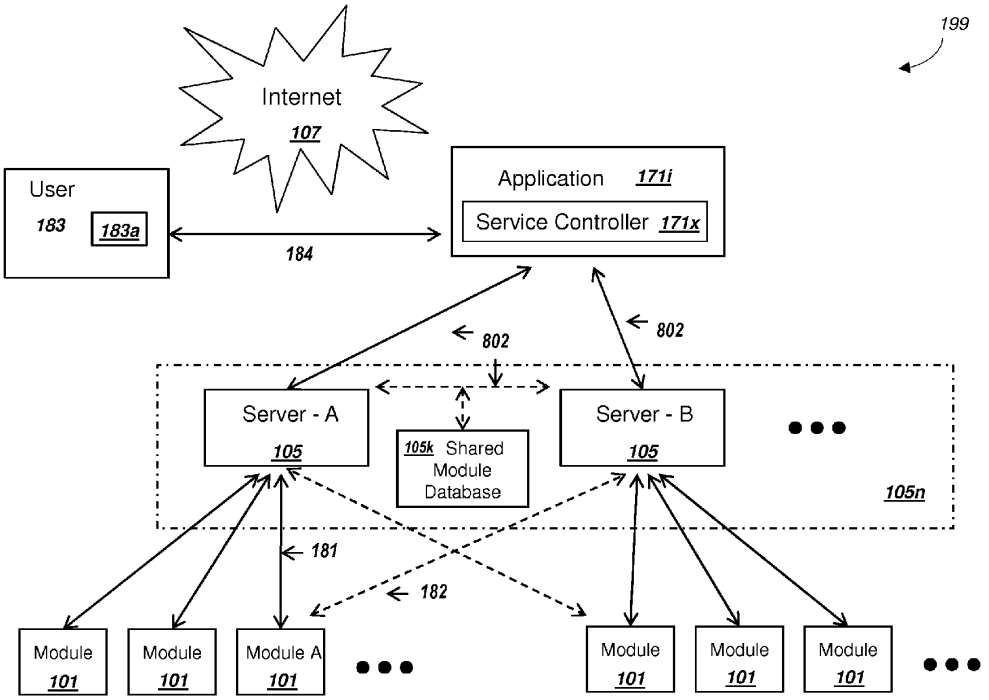
*H04L 9/00* (2006.01)

(52) **U.S. Cl.**

CPC ..... ***H04L 9/321*** (2013.01); ***H04L 9/3247*** (2013.01); ***H04L 9/006*** (2013.01)

(57) **ABSTRACT**

A set of servers can support secure and efficient "Machine to Machine" communications using an application interface and a module controller. The set of servers can record data for a plurality of modules in a shared module database. The set of servers can (i) access the Internet to communicate with a module using a module identity, (i) receive server instructions, and (iii) send module instructions. Data can be encrypted and decrypted using a set of cryptographic algorithms and a set of cryptographic parameters. The set of servers can (i) receive a module public key with a module identity, (ii) authenticate the module public key, and (iii) receive a subsequent series of module public keys derived by the module with a module identity. The application interface can use a first server private key and the module controller can use a second server private key.



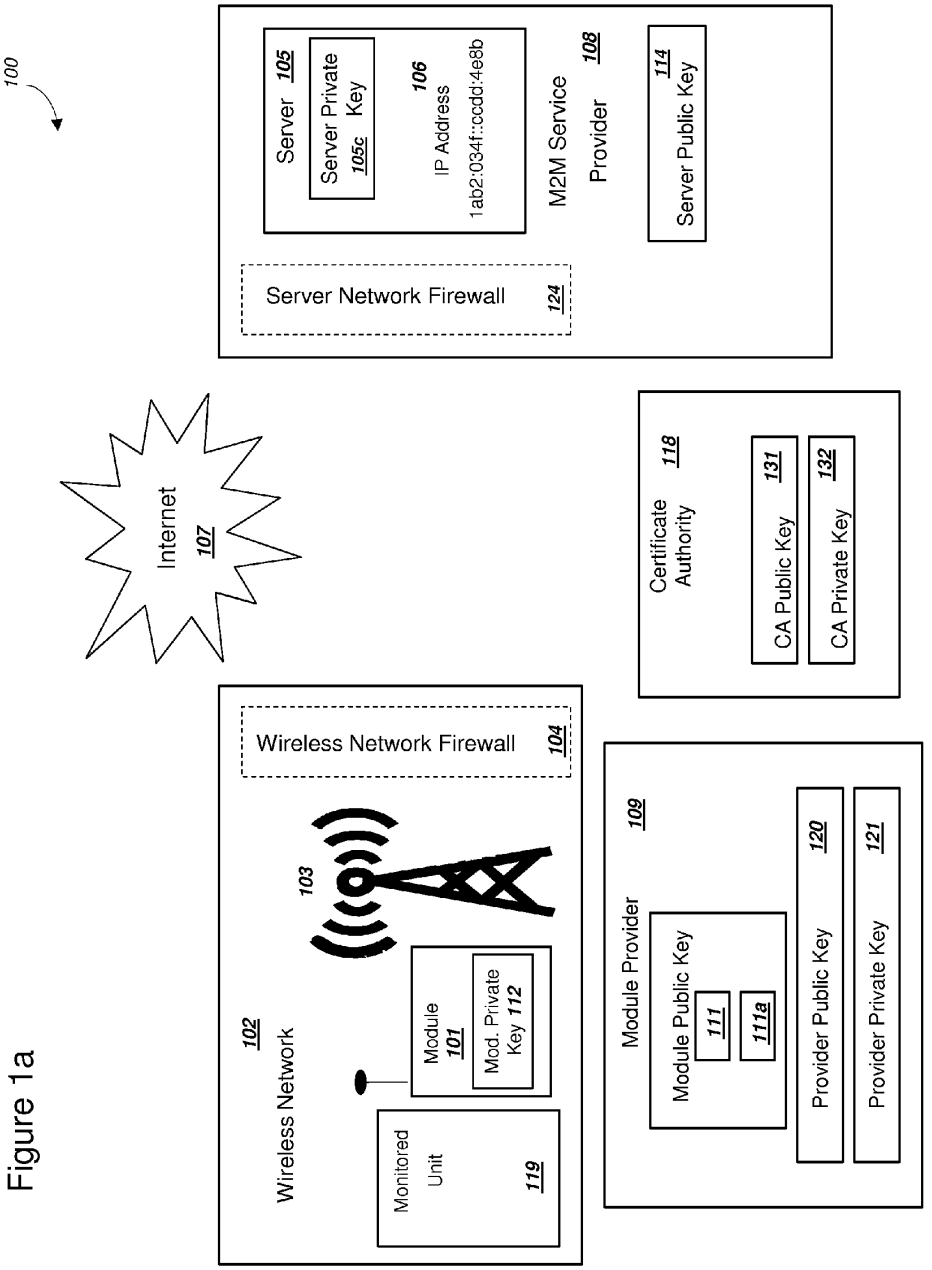

100
Server 105
Server Private Key 105c
106
IP Address
1ab2:034f::ccdd:4e8b
M2M Service Provider 108
114
Server Public Key
Server Network Firewall 124
Internet 107
Certificate Authority 118
CA Public Key 131
CA Private Key 132
Wireless Network Firewall 104
103
102
Wireless Network
Module 101
Mod. Private Key 112
Monitored Unit 119
Module Provider 109
Module Public Key
111
111a
Provider Public Key 120
Provider Private Key 121

Figure 1a

Figure 1d

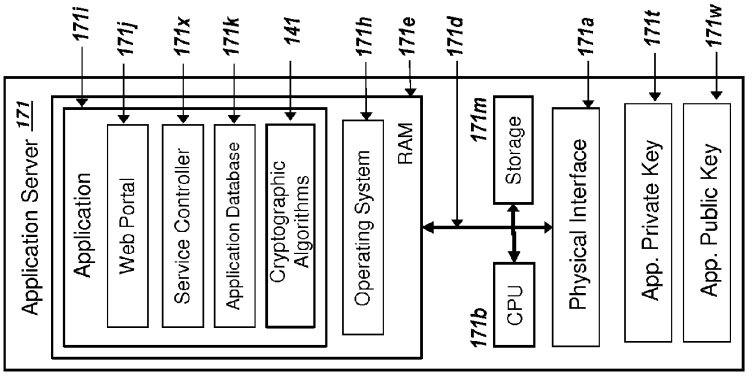



Figure 1c

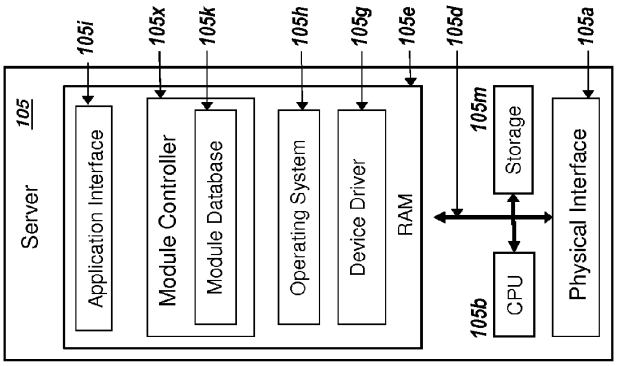



Figure 1b

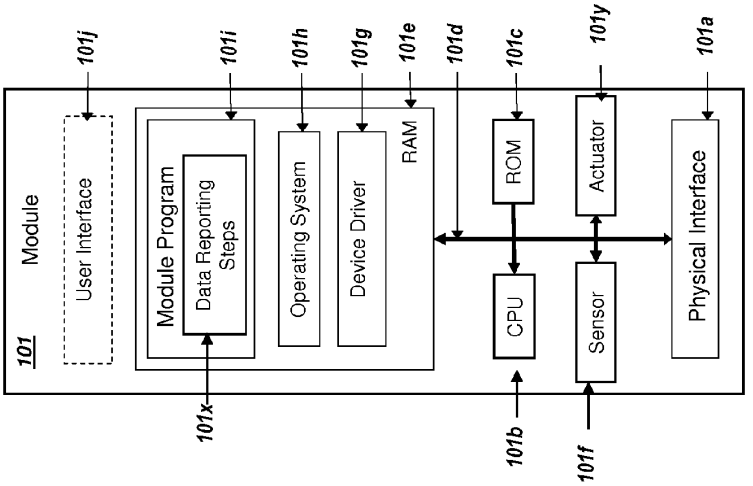


Figure 1f

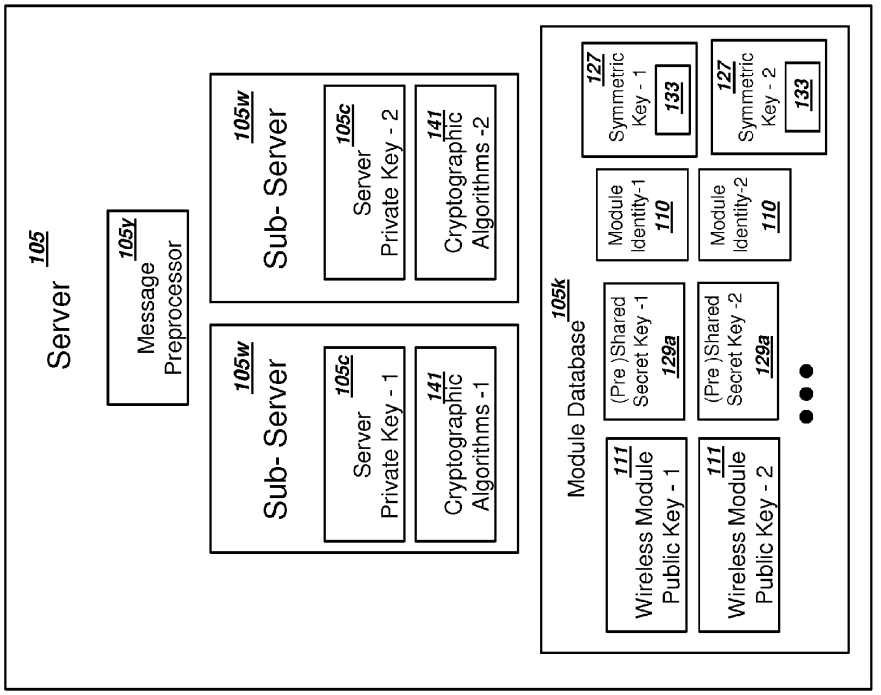
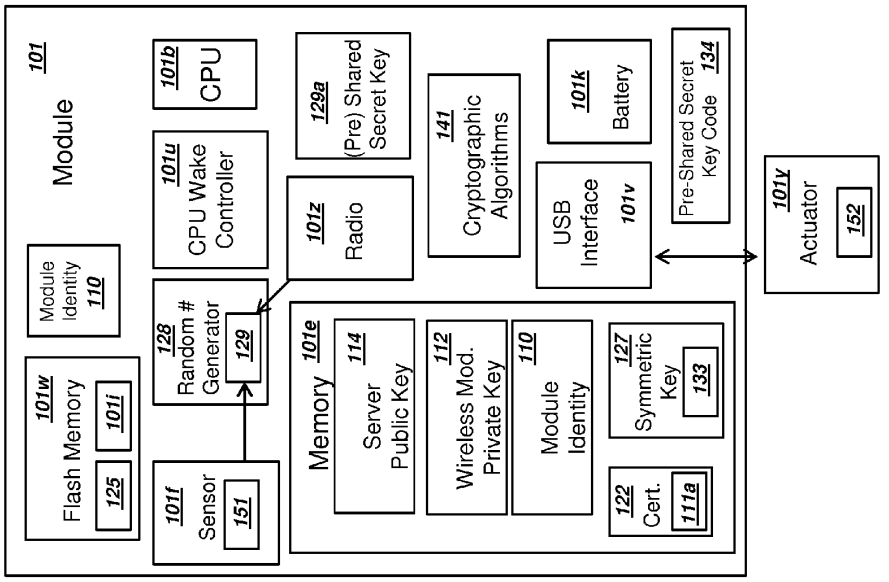



Figure 1e



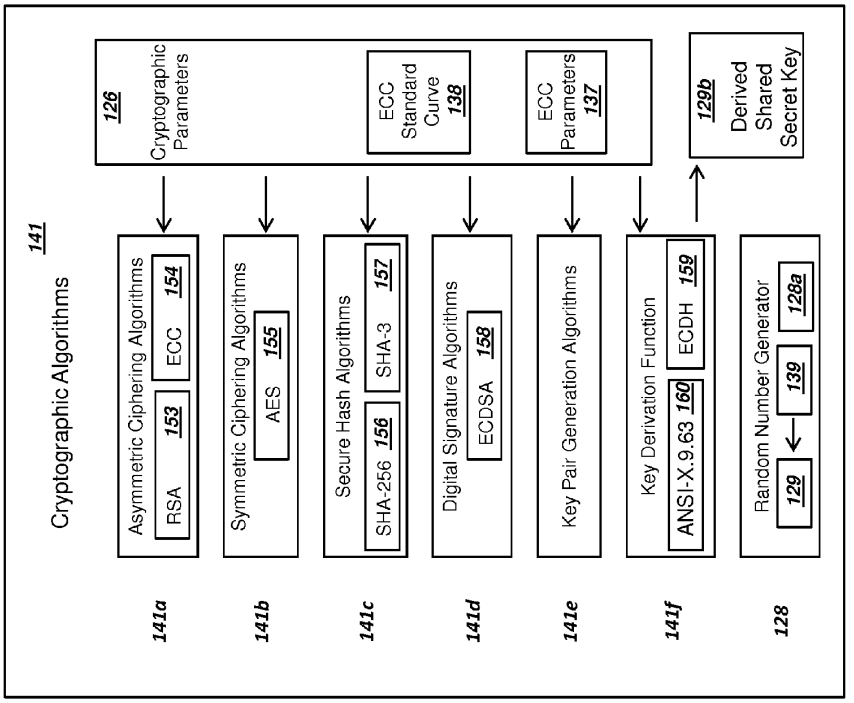

141
Cryptographic Algorithms
126
Cryptographic Parameters
141a
Asymmetric Ciphering Algorithms
RSA 153
ECC 154
141b
Symmetric Ciphering Algorithms
AES 155
141c
Secure Hash Algorithms
SHA-256 156
SHA-3 157
ECC Standard Curve 138
141d
Digital Signature Algorithms
ECDSA 158
ECC Parameters 137
141e
Key Pair Generation Algorithms
141f
Key Derivation Function
ANSI-X.9.63 160
ECDH 159
129b
Derived Shared Secret Key
128
Random Number Generator
129
139
128a

Figure 1g

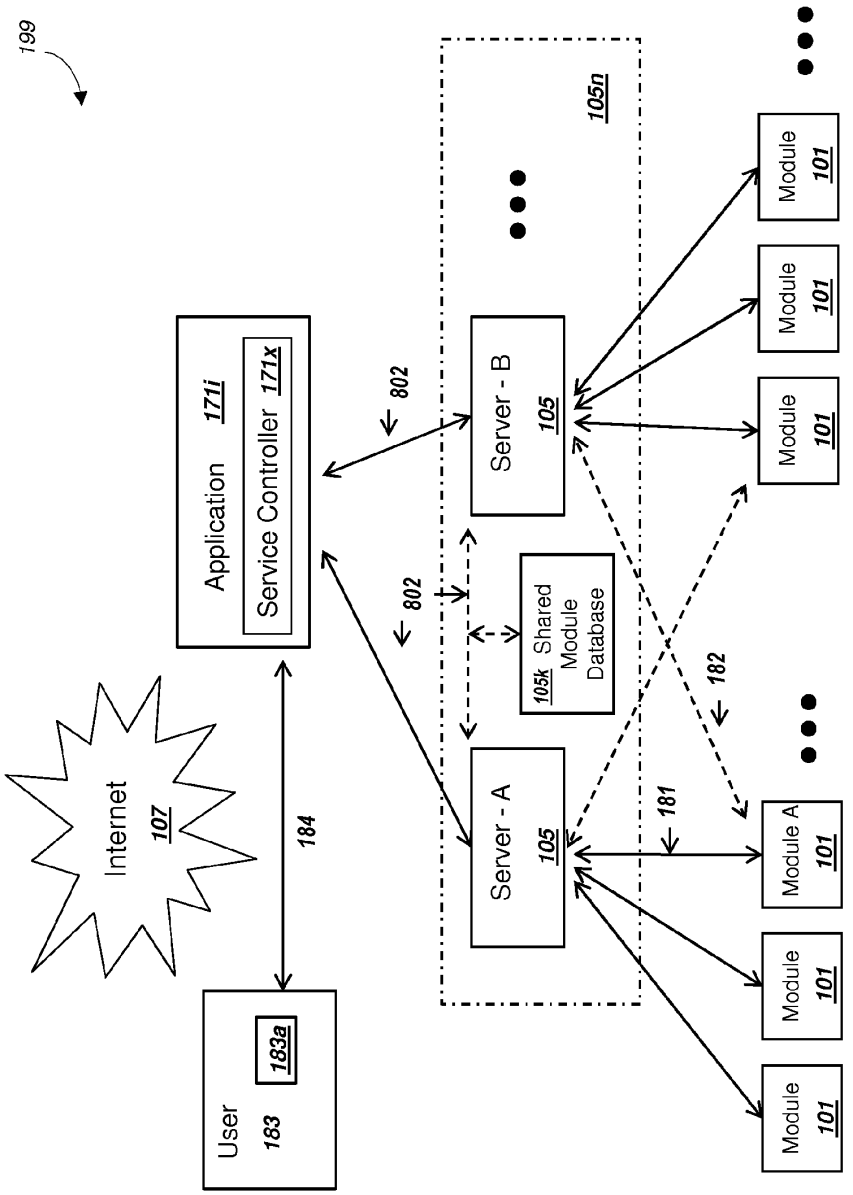
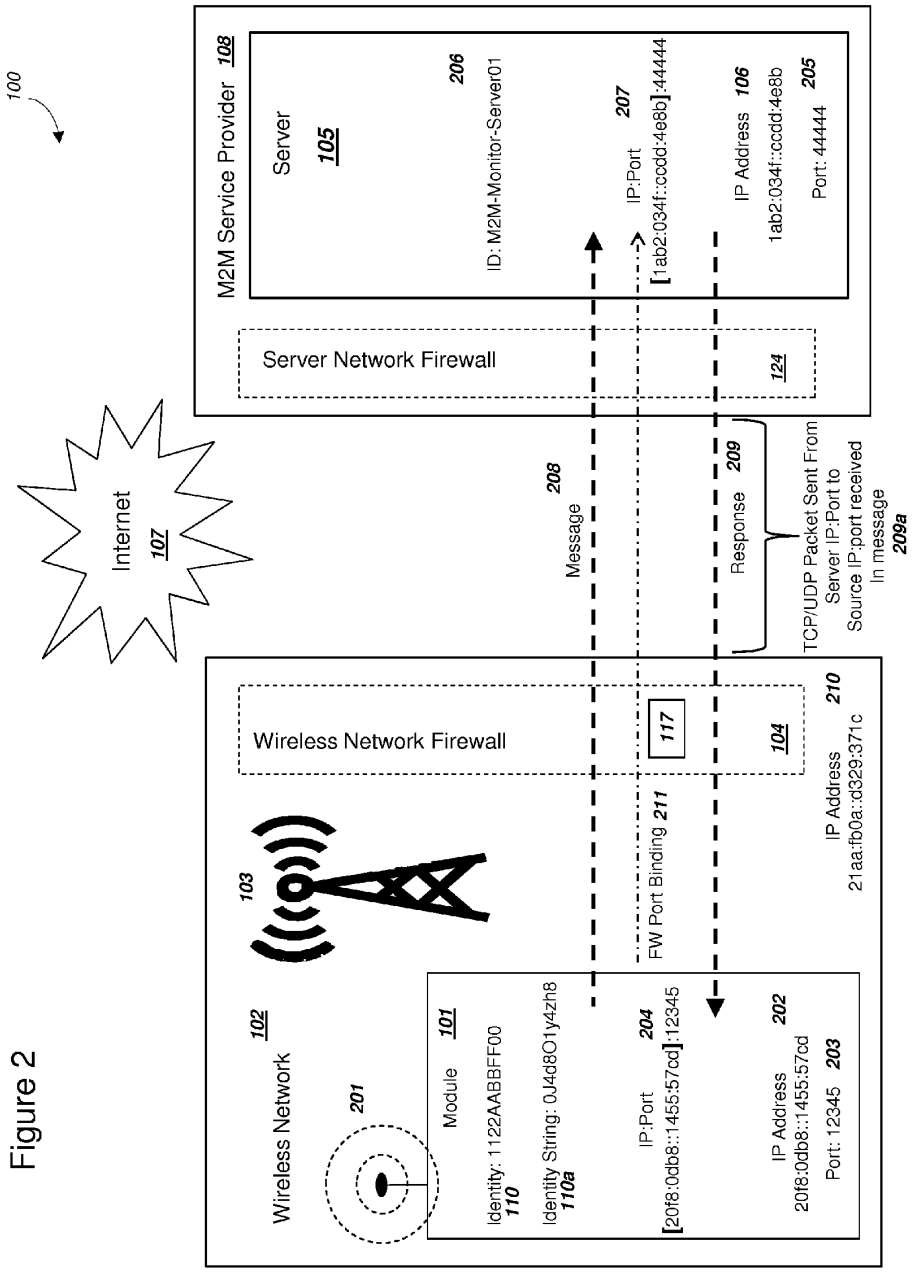

199
Internet
107
Application
171i
Service Controller
171x
User
183
183a
184
802
802
Server - A
105
Server - B
105
105n
105k
Shared
Module
Database
181
182
Module A
101
Module
101
Module
101
Module
101
Module
101

Figure 1h

Figure 2


100
M2M Service Provider 108
Server
105
206
ID: M2M-Monitor-Server01
207
IP:Port
[1ab2:034f::ccdd:4e8b]:44444
IP Address 106
1ab2:034f::ccdd:4e8b
205
Port: 44444
Server Network Firewall
124
Internet
107
208
Message
209
Response
TCP/UDP Packet Sent From Server IP:Port to Source IP:port received In message
209a
Wireless Network Firewall
117
104
210
IP Address
21aa:fb0a::d329:371c
103
FW Port Binding 211
102
Wireless Network
201
Module 101
Identity: 1122AABBFF00
110
Identity String: 0J4d8O1y4zh8
110a
204
IP:Port
[20f8:0db8::1455:57cd]:12345
202
IP Address
20f8:0db8::1455:57cd
Port: 12345 203

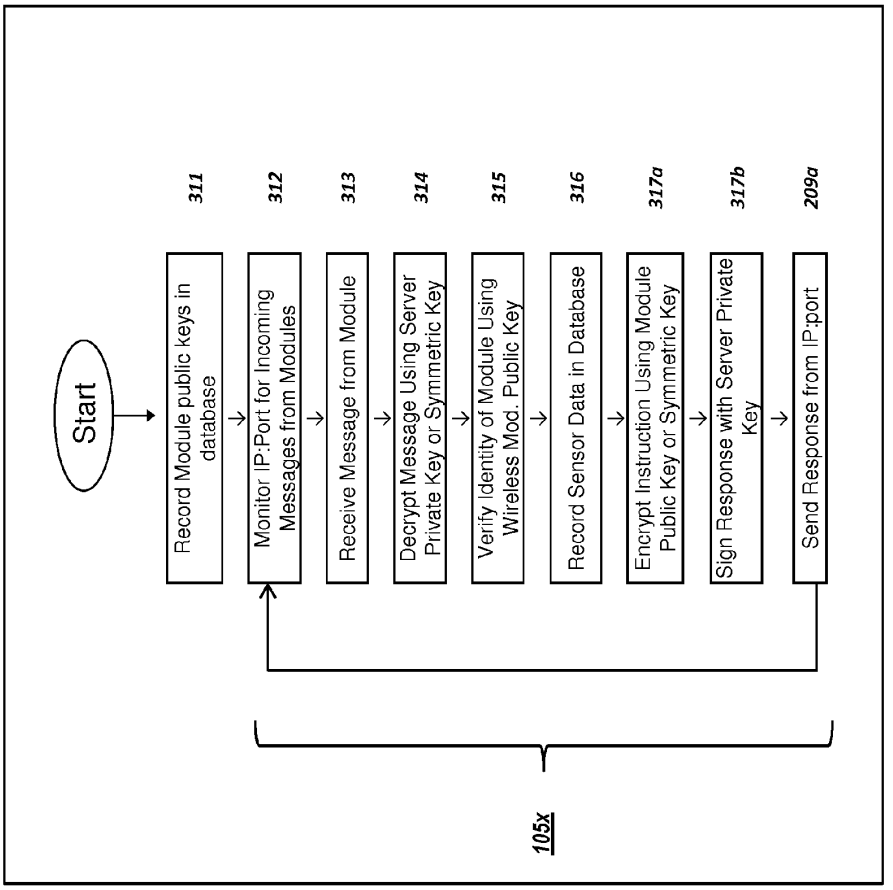

Start
Record Module public keys in database
311
Monitor IP:Port for Incoming Messages from Modules
312
Receive Message from Module
313
Decrypt Message Using Server Private Key or Symmetric Key
314
Verify Identity of Module Using Wireless Mod. Public Key
315
Record Sensor Data in Database
316
Encrypt Instruction Using Module Public Key or Symmetric Key
317a
Sign Response with Server Private Key
317b
Send Response from IP:port
209a
105x

Figure 3

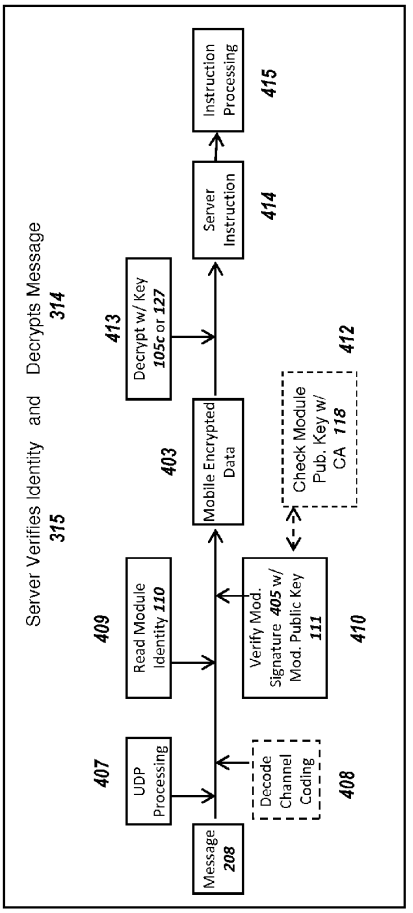



Figure 4

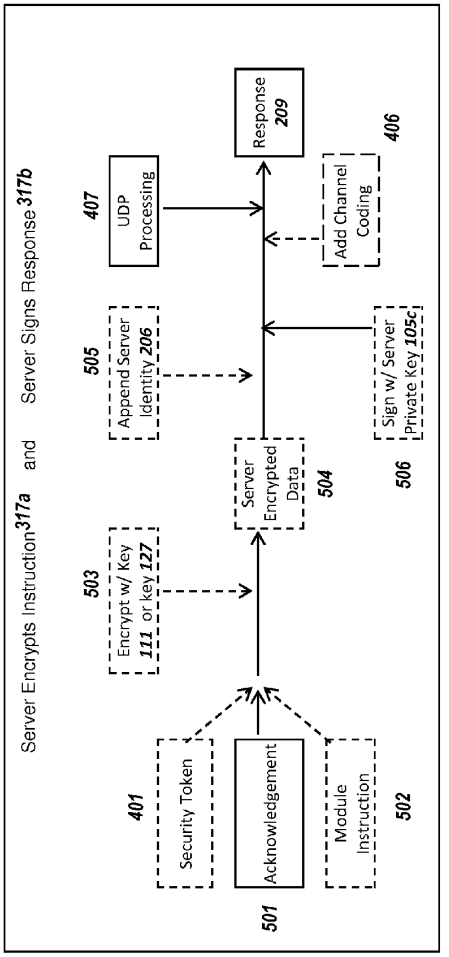



Figure 5a

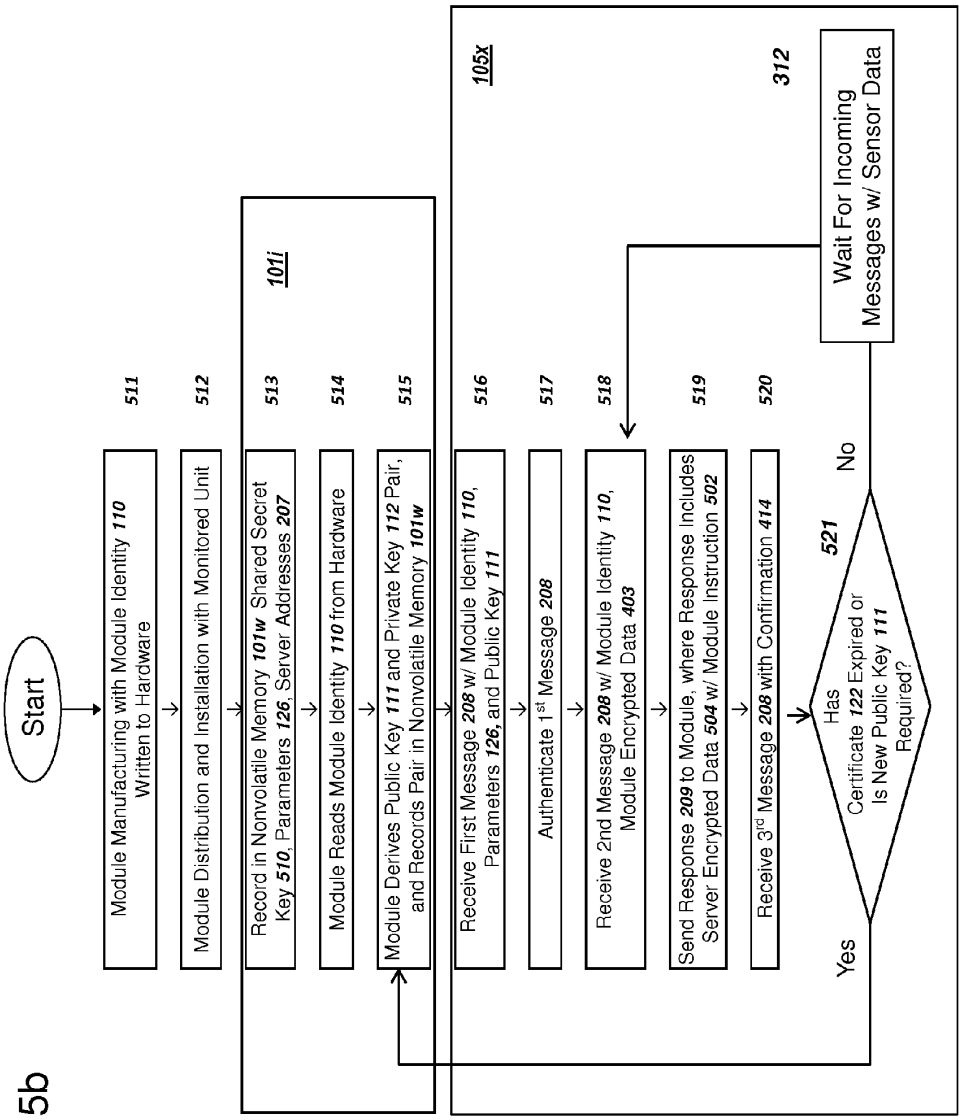

Start
Module Manufacturing with Module Identity 110 Written to Hardware
511
Module Distribution and Installation with Monitored Unit
512
Record in Nonvolatile Memory 101w Shared Secret Key 510, Parameters 126, Server Addresses 207
513
101i
Module Reads Module Identity 110 from Hardware
514
Module Derives Public Key 111 and Private Key 112 Pair, and Records Pair in Nonvolatile Memory 101w
515
105x
Receive First Message 208 w/ Module Identity 110, Parameters 126, and Public Key 111
516
Authenticate 1st Message 208
517
Receive 2nd Message 208 w/ Module Identity 110, Module Encrypted Data 403
518
Send Response 209 to Module, where Response Includes Server Encrypted Data 504 w/ Module Instruction 502
519
Receive 3rd Message 208 with Confirmation 414
520
Has Certificate 122 Expired or Is New Public Key 111 Required?
521
Yes
No
Wait For Incoming Messages w/ Sensor Data
312

Figure 5b

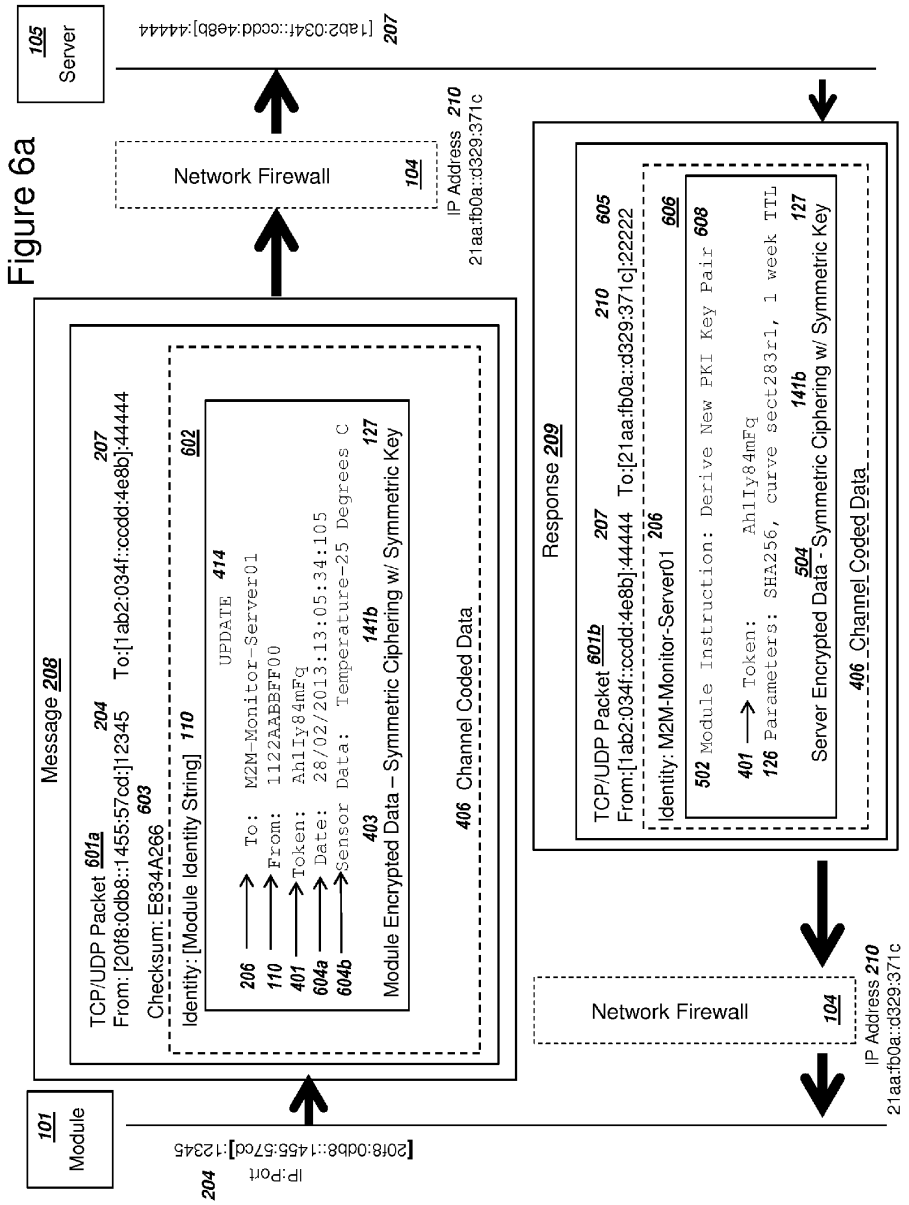
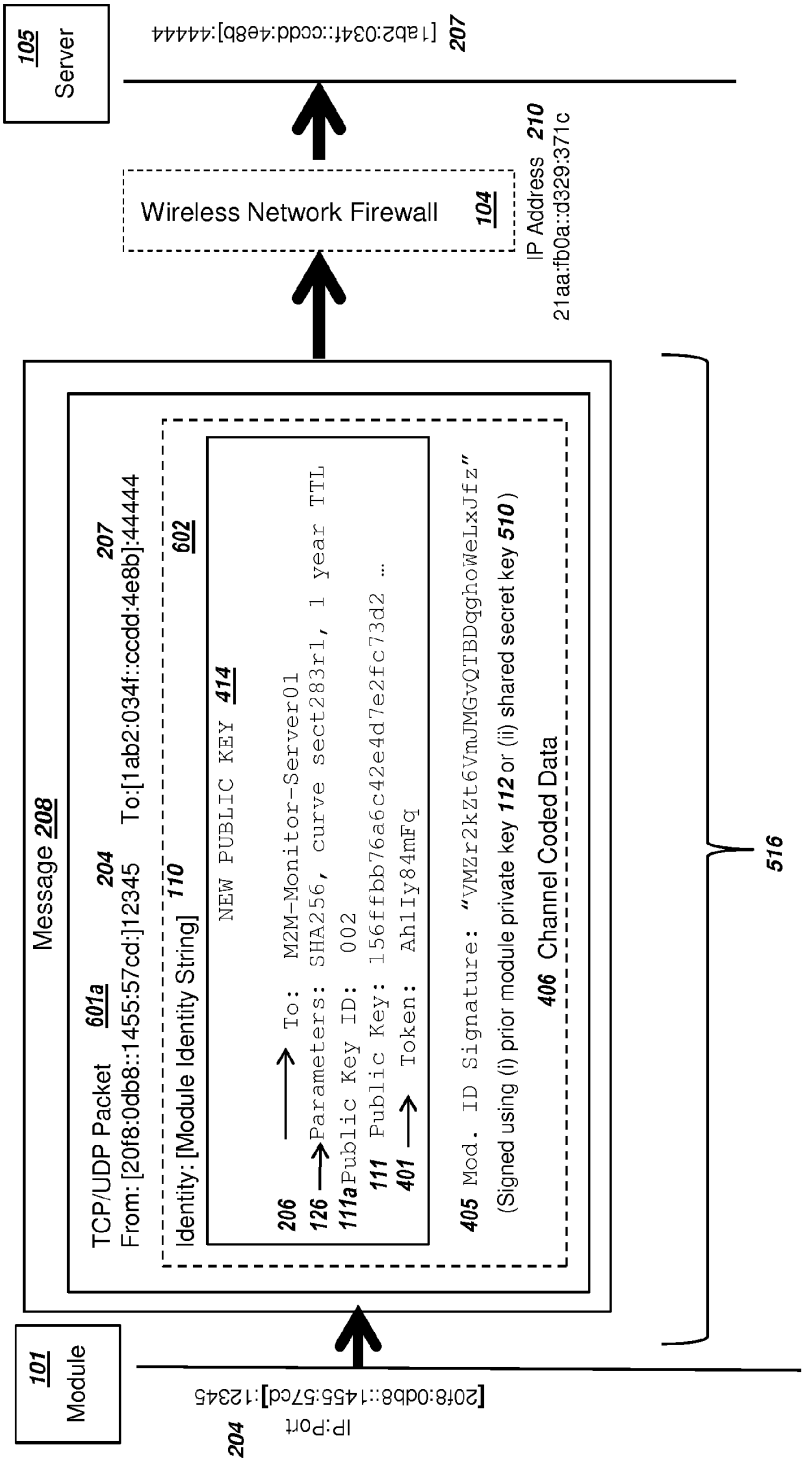


Figure 6a
101 Module
105 Server
Message 208
TCP/UDP Packet 601a
From: [20f8:0db8::1455:57cd]:12345 204
To:[1ab2:034f::ccdd:4e8b]:44444 207
Checksum: E834A266 603
Identity: [Module Identity String] 110
602
UPDATE 414
206 To: M2M-Monitor-Server01
110 From: 1122AABBFF00
401 Token: AhlIy84mFq
604a Date: 28/02/2013:13:05:34:105
604b Sensor Data: Temperature-25 Degrees C
127
403
Module Encrypted Data – Symmetric Ciphering w/ Symmetric Key 141b
406 Channel Coded Data
Network Firewall 104
IP Address 210
21aa:fb0a::d329:371c
IP:Port
[20f8:0db8::1455:57cd]:12345
204
[1ab2:034f::ccdd:4e8b]:44444 207
Response 209
TCP/UDP Packet 601b 207
From:[1ab2:034f::ccdd:4e8b]:44444 To:[21aa:fb0a::d329:371c]:22222 210 605
Identity: M2M-Monitor-Server01 206
606
502 Module Instruction: Derive New PKI Key Pair 608
401 Token: AhlIy84mFq
126 Parameters: SHA256, curve sect283r1, 1 week TTL
504
Server Encrypted Data - Symmetric Ciphering w/ Symmetric Key 141b 127
406 Channel Coded Data
Network Firewall 104
IP Address 210
21aa:fb0a::d329:371c

Figure 6b


105
Server
[1ab2:034f::ccdd:4e8b]:44444
207
Wireless Network Firewall
104
IP Address 210
21aa:fb0a::d329:371c
Message 208
TCP/UDP Packet 601a
From: [20f8:0db8::1455:57cd:]12345
204
To:[1ab2:034f::ccdd:4e8b]:44444
207
Identity: [Module Identity String] 110
602
NEW PUBLIC KEY 414
206 To: M2M-Monitor-Server01
126 Parameters: SHA256, curve sect283r1, 1 year TTL
111a Public Key ID: 002
111 Public Key: 156ffbb76a6c42e4d7e2fc73d2 …
401 Token: Ah1Iy84mFq
405 Mod. ID Signature: "VMZr2kZt6VmJMGvQIBDqghoWeLxJfz"
(Signed using (i) prior module private key 112 or (ii) shared secret key 510)
406 Channel Coded Data
516
101
Module
IP:Port
[20f8:0db8::1455:57cd]:12345
204

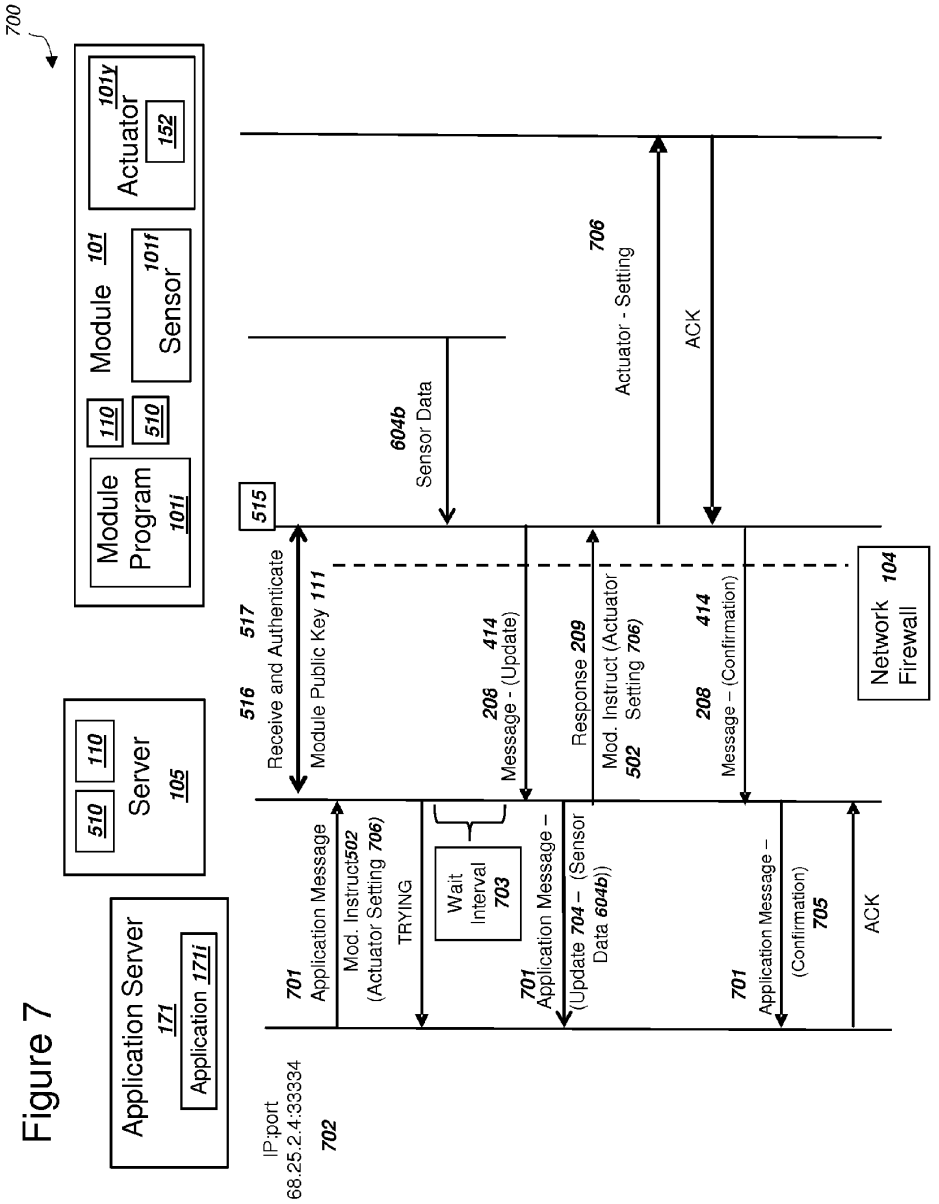

700
Module 101
Module Program 101i
110
510
Sensor 101f
Actuator 101y
152
Server 105
510
110
Application Server 171
Application 171i
IP:port 68.25.2.4:33334 702
515
516 Receive and Authenticate
517 Module Public Key 111
604b Sensor Data
701 Application Message
Mod. Instruct 502 (Actuator Setting 706)
TRYING
Wait Interval 703
208 Message - (Update)
414
701 Application Message – (Update 704 – (Sensor Data 604b))
Response 209
Mod. Instruct (Actuator Setting 706) 502
706 Actuator - Setting
ACK
208 Message – (Confirmation)
414
701 Application Message – (Confirmation) 705
ACK
Network Firewall 104

Figure 7

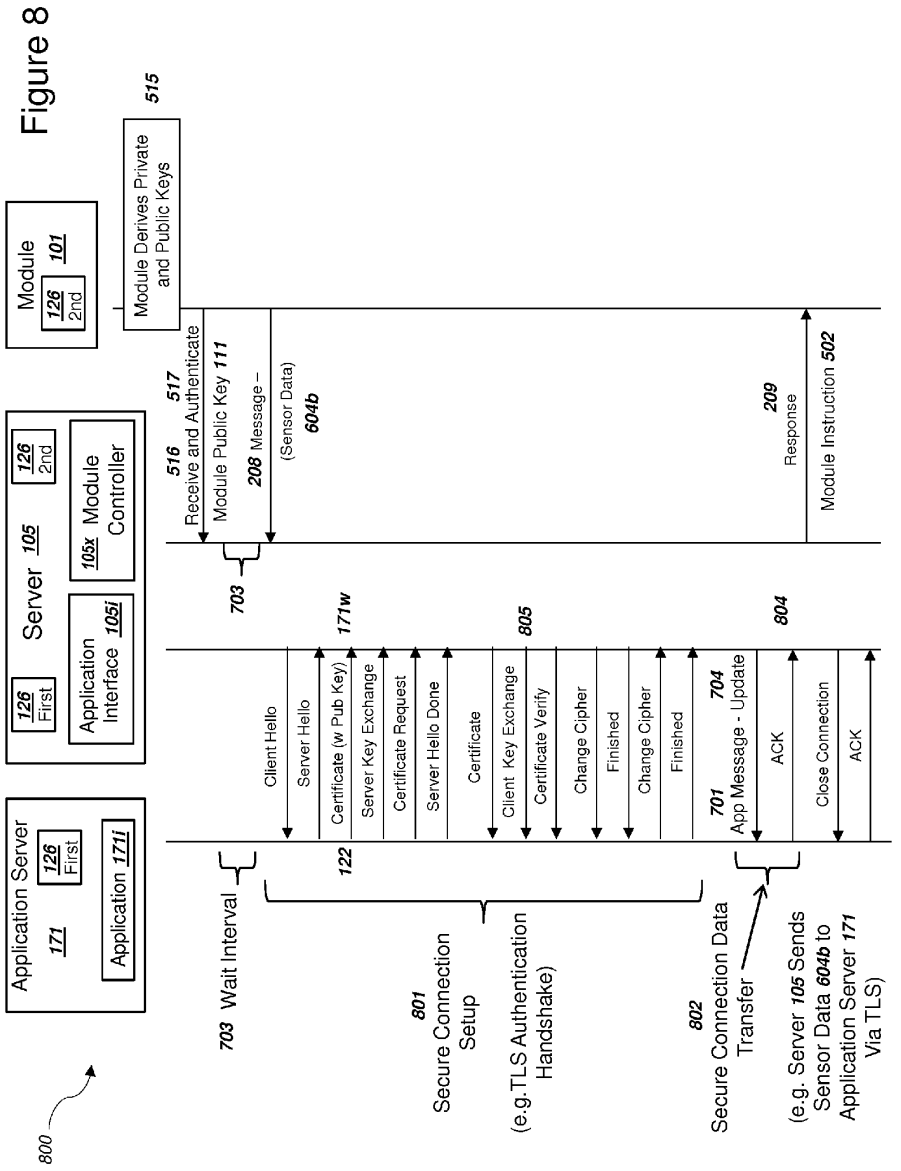

800
Application Server
171
126 First
Application 171i
Server 105
126 First
Application Interface 105i
105x Module Controller
126 2nd
Module
126 2nd
101
515
Module Derives Private and Public Keys
517
516 Receive and Authenticate
Module Public Key 111
208 Message –
(Sensor Data)
604b
209
Response
Module Instruction 502
703
171w
805
804
Client Hello
Server Hello
Certificate (w Pub Key)
Server Key Exchange
Certificate Request
Server Hello Done
Certificate
Client Key Exchange
Certificate Verify
Change Cipher
Finished
Change Cipher
Finished
701
App Message - Update
704
ACK
Close Connection
ACK
122
703 Wait Interval
801
Secure Connection Setup
(e.g.TLS Authentication Handshake)
802
Secure Connection Data Transfer
(e.g. Server 105 Sends Sensor Data 604b to Application Server 171 Via TLS)

Figure 8

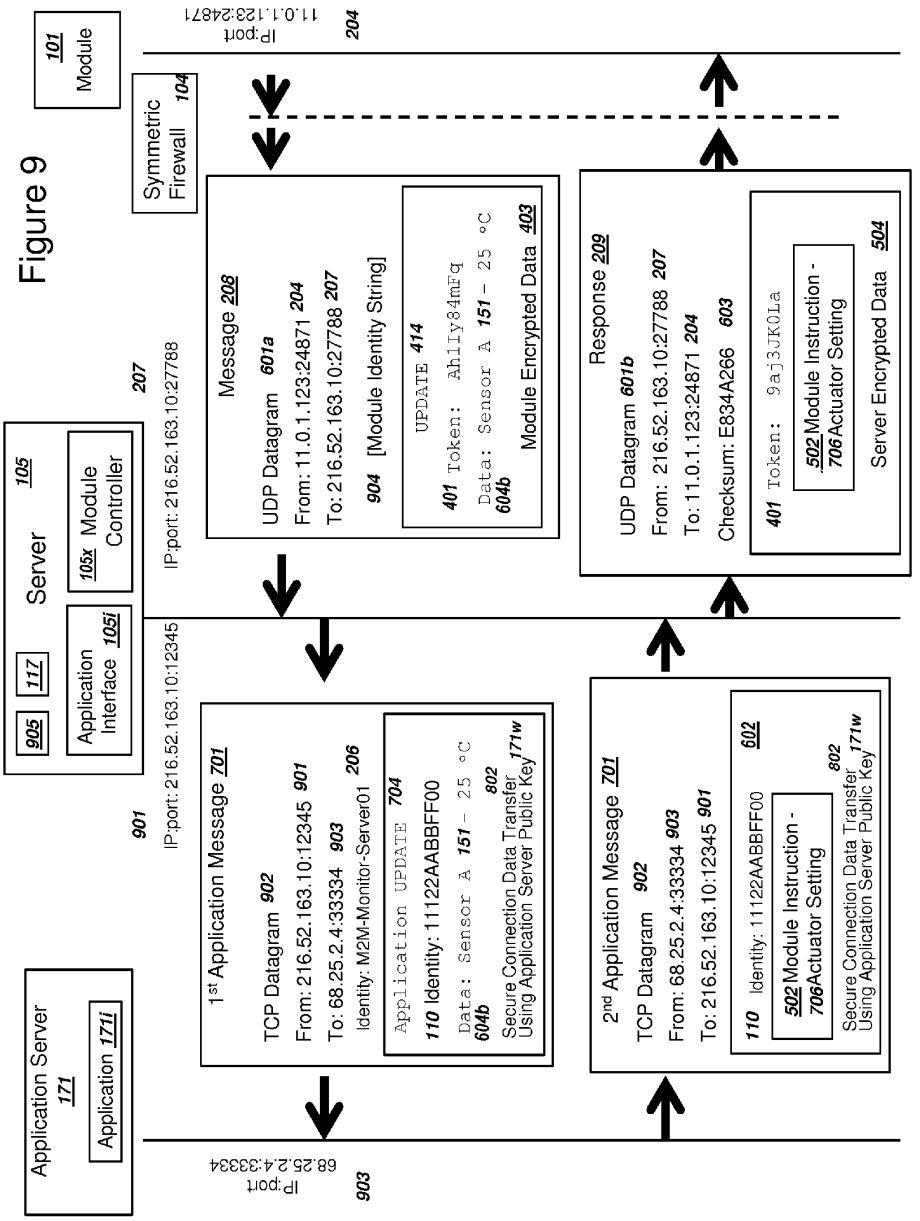

Application Server 171
Application 171i
905
117
Application Interface 105i
Server 105
105x Module Controller
901
IP:port: 216.52.163.10:12345
1st Application Message 701
TCP Datagram 902
From: 216.52.163.10:12345 901
To: 68.25.2.4:33334 903
Identity: M2M-Monitor-Server01 206
Application UPDATE 704
110 Identity: 11122AABBFF00
Data: Sensor A 151 - 25 °C
604b
802
Secure Connection Data Transfer Using Application Server Public Key 171w
IP:port
68.25.2.4:33334
903
207
IP:port: 216.52.163.10:27788
2nd Application Message 701
TCP Datagram 902
From: 68.25.2.4:33334 903
To: 216.52.163.10:12345 901
110 Identity: 11122AABBFF00 602
502 Module Instruction - 706 Actuator Setting
802
Secure Connection Data Transfer Using Application Server Public Key 171w
Symmetric Firewall 104
Message 208
UDP Datagram 601a
From: 11.0.1.123:24871 204
To: 216.52.163.10:27788 207
904 [Module Identity String]
UPDATE 414
401 Token: Ahlly84mFq
Data: Sensor A 151 - 25 °C
604b
Module Encrypted Data 403
Response 209
UDP Datagram 601b
From: 216.52.163.10:27788 207
To: 11.0.1.123:24871 204
Checksum: E834A266 603
401 Token: 9aj3JK0La
502 Module Instruction - 706 Actuator Setting
Server Encrypted Data 504
IP:port
11.0.1.123:24871
204
101 Module

Figure 9

Figure 10

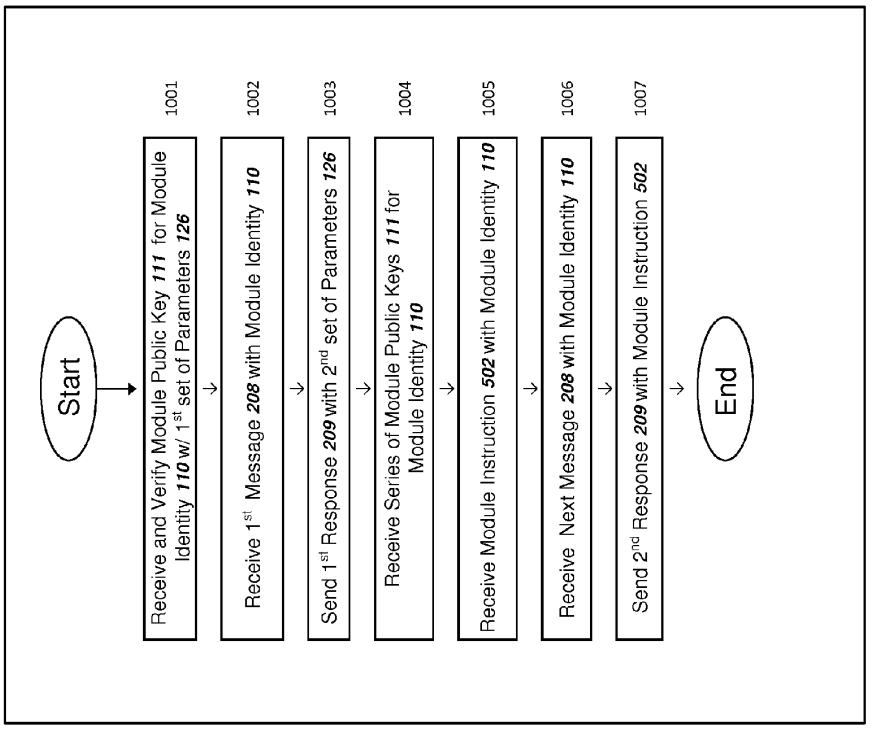
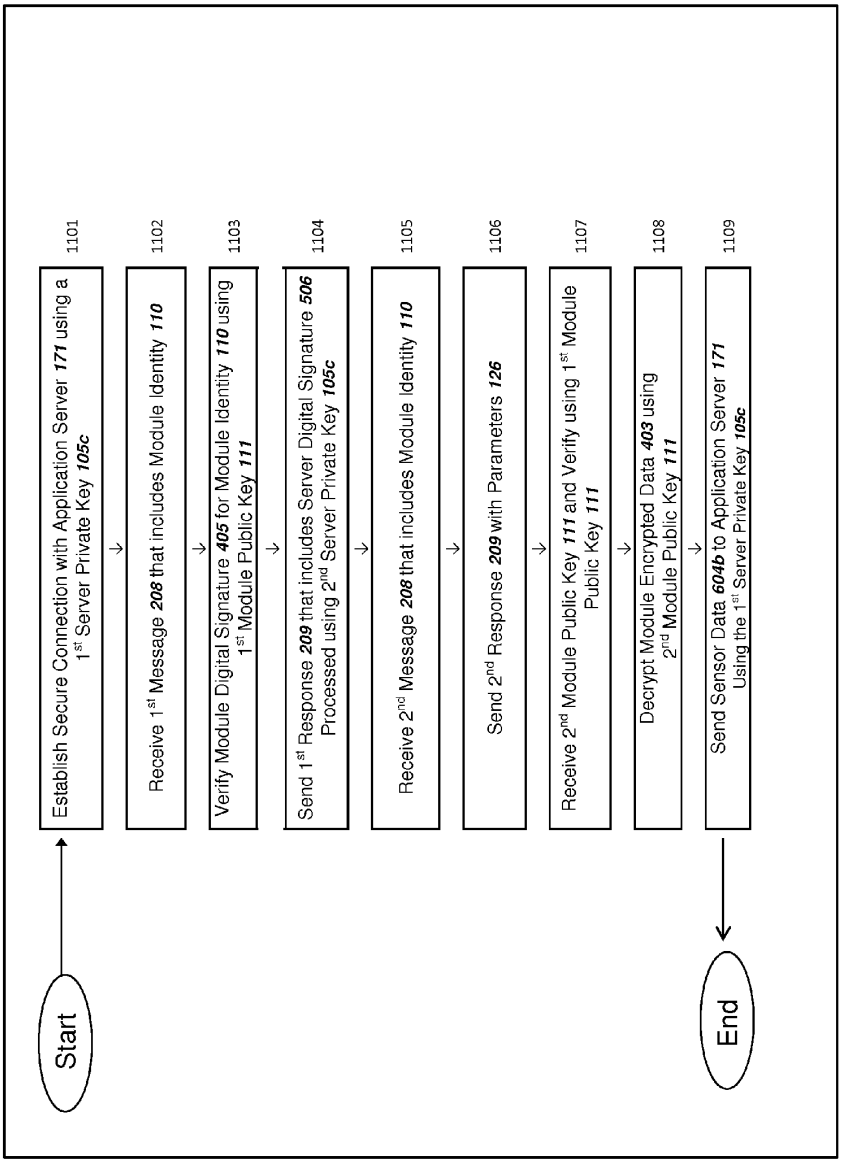

Start
Receive and Verify Module Public Key 111 for Module Identity 110 w/ 1st set of Parameters 126
1001
Receive 1st Message 208 with Module Identity 110
1002
Send 1st Response 209 with 2nd set of Parameters 126
1003
Receive Series of Module Public Keys 111 for Module Identity 110
1004
Receive Module Instruction 502 with Module Identity 110
1005
Receive Next Message 208 with Module Identity 110
1006
Send 2nd Response 209 with Module Instruction 502
1007
End

Figure 11


Start
Establish Secure Connection with Application Server 171 using a 1st Server Private Key 105c
1101
Receive 1st Message 208 that includes Module Identity 110
1102
Verify Module Digital Signature 405 for Module Identity 110 using 1st Module Public Key 111
1103
Send 1st Response 209 that includes Server Digital Signature 506 Processed using 2nd Server Private Key 105c
1104
Receive 2nd Message 208 that includes Module Identity 110
1105
Send 2nd Response 209 with Parameters 126
1106
Receive 2nd Module Public Key 111 and Verify using 1st Module Public Key 111
1107
Decrypt Module Encrypted Data 403 using 2nd Module Public Key 111
1108
Send Sensor Data 604b to Application Server 171 Using the 1st Server Private Key 105c
1109
End

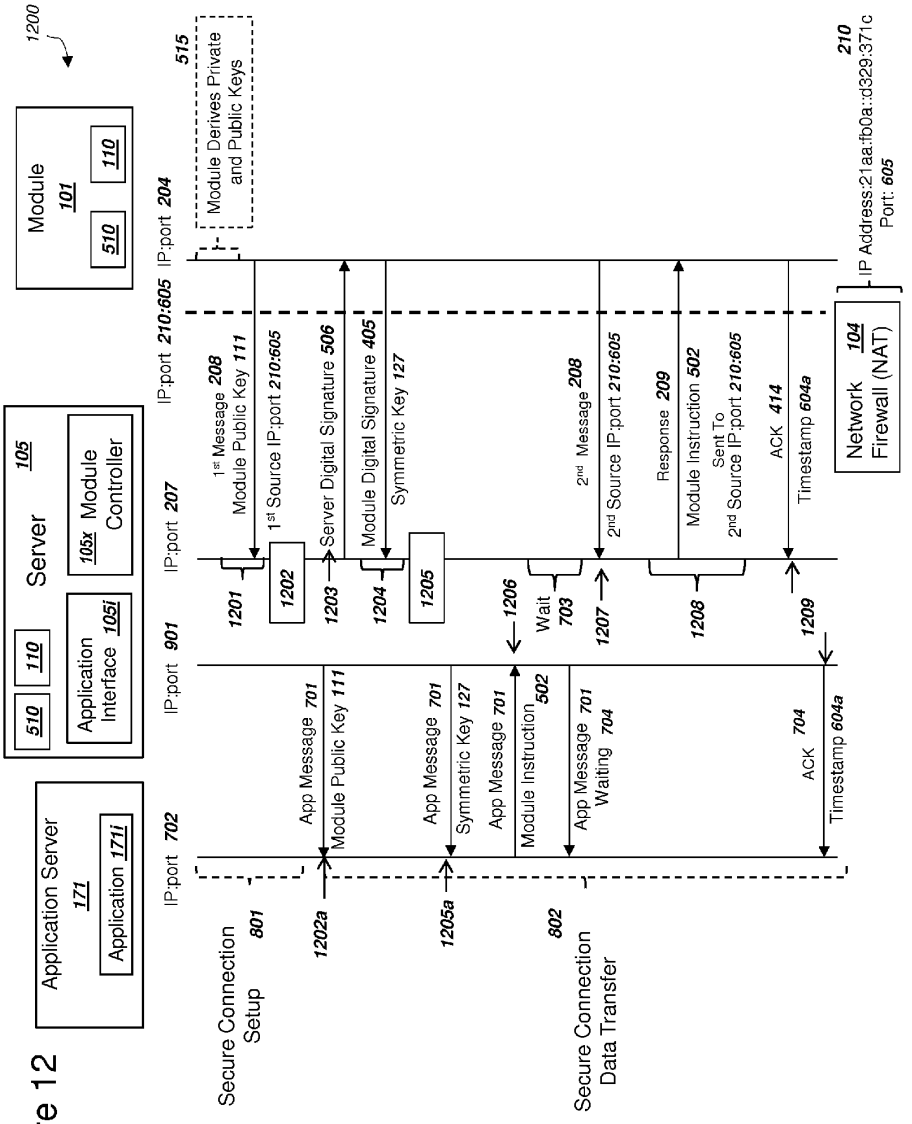

1200
Module
101
510
110
Module Derives Private and Public Keys
515
IP:port 204
IP:port 210:605
IP:port 207
IP:port 901
IP:port 702
Server
105
105x Module Controller
Application Interface 105i
Application Server
171
Application 171i
1st Message 208
Module Public Key 111
1st Source IP:port 210:605
Server Digital Signature 506
Module Digital Signature 405
Symmetric Key 127
2nd Message 208
2nd Source IP:port 210:605
Response 209
Module Instruction 502
Sent To
2nd Source IP:port 210:605
ACK 414
Timestamp 604a
Network 104
Firewall (NAT)
IP Address:21aa:fb0a::d329:371c
Port: 605
210
App Message 701
Module Public Key 111
App Message 701
Symmetric Key 127
App Message 701
Module Instruction 502
App Message 701
Waiting 704
ACK 704
Timestamp 604a
1201
1202
1203
1204
1205
1206
Wait
703
1207
1208
1209
1202a
1205a
Secure Connection Setup
801
802
Secure Connection Data Transfer

Figure 12

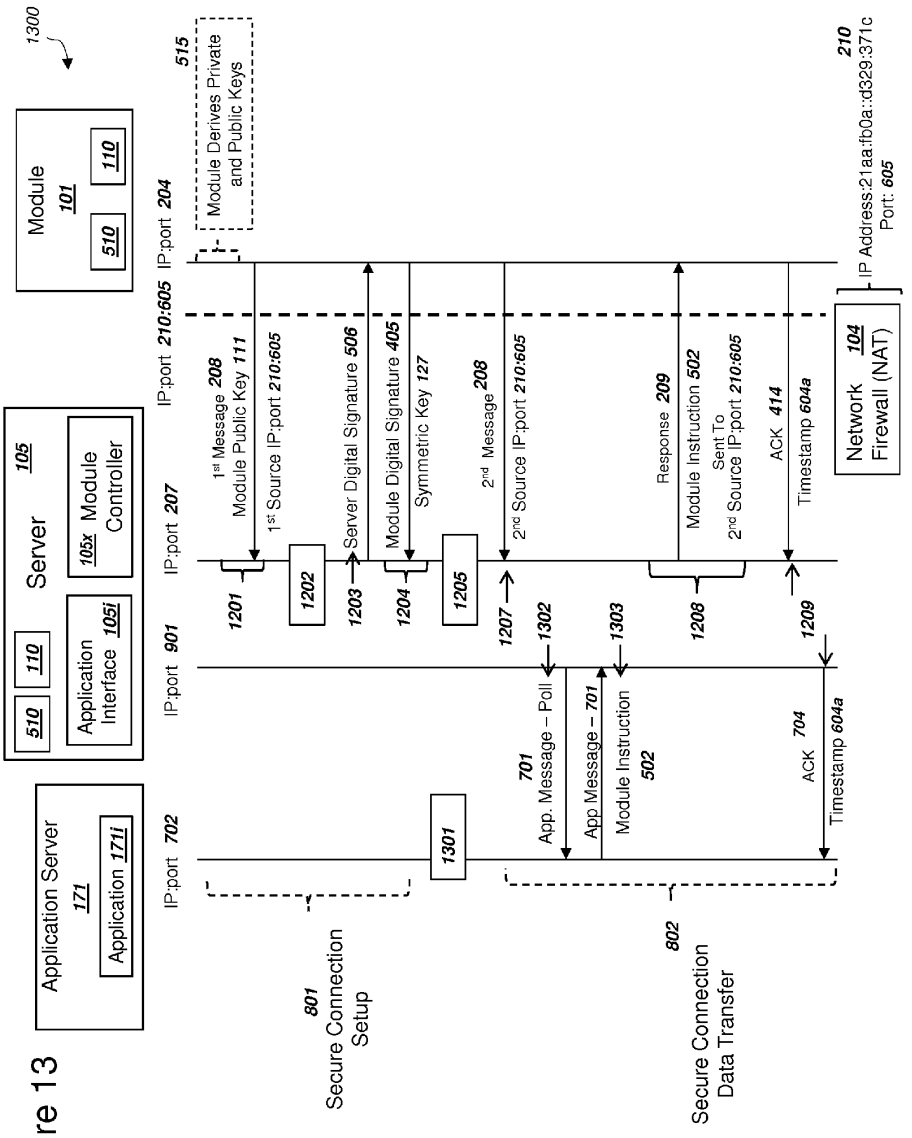

1300
Module
101
510
110
Server
105
105x Module Controller
Application Interface 105i
Application Server
171
Application 171i
IP:port 702
IP:port 901
IP:port 207
IP:port 210:605
IP:port 204
515
Module Derives Private and Public Keys
1st Message 208
Module Public Key 111
1st Source IP:port 210:605
Server Digital Signature 506
Module Digital Signature 405
Symmetric Key 127
2nd Message 208
2nd Source IP:port 210:605
Response 209
Module Instruction 502
Sent To
2nd Source IP:port 210:605
ACK 414
Timestamp 604a
Network 104
Firewall (NAT)
210
IP Address:21aa:fb0a::d329:371c
Port: 605
1201
1202
1203
1204
1205
1207
1208
1209
1301
1302
1303
701
App. Message – Poll
App Message – 701
Module Instruction
502
ACK 704
Timestamp 604a
801
Secure Connection Setup
802
Secure Connection Data Transfer

Figure 13

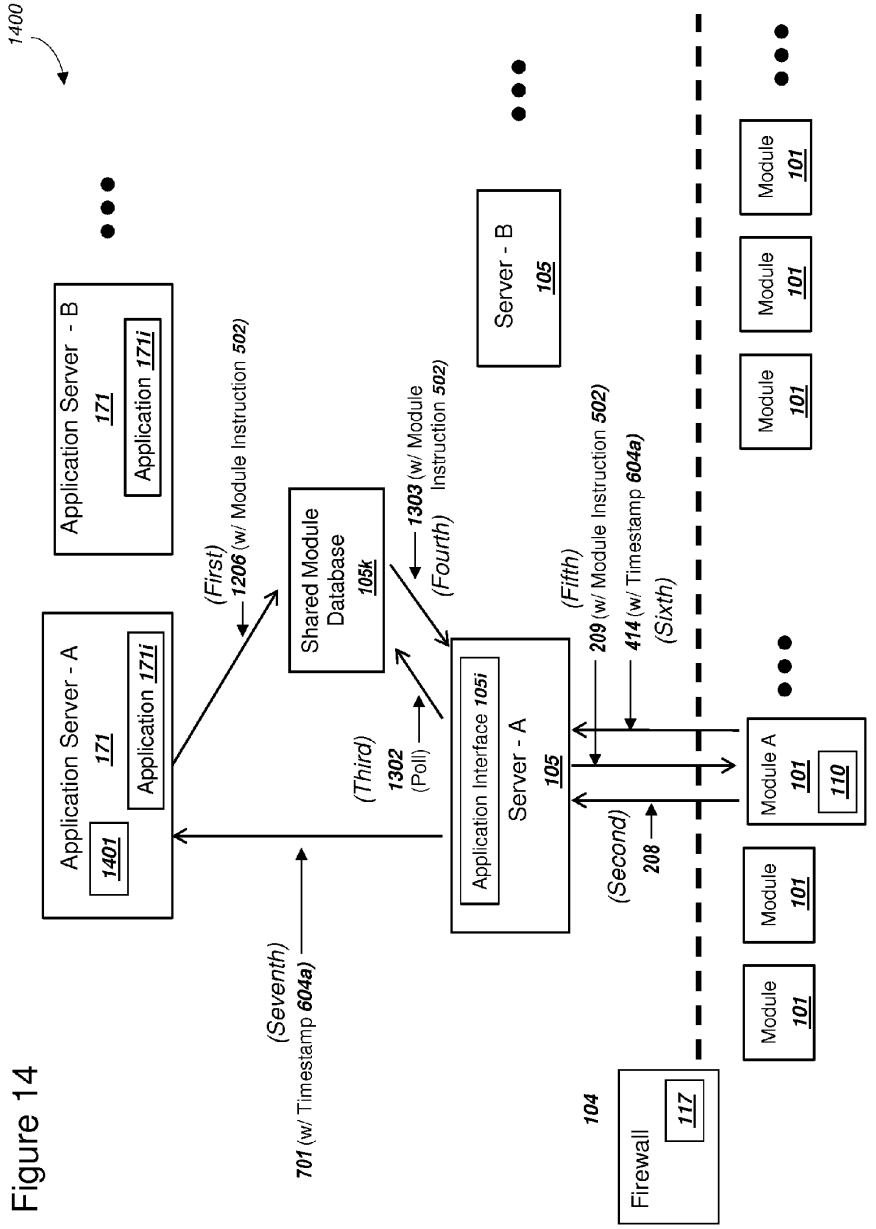

1400
Application Server - A
171
1401
Application 171i
Application Server - B
171
Application 171i
(First)
1206 (w/ Module Instruction 502)
Shared Module Database
105k
(Third)
1302
(Poll)
1303 (w/ Module Instruction 502)
(Fourth)
Application Interface 105i
Server - A
105
Server - B
105
(Fifth)
209 (w/ Module Instruction 502)
414 (w/ Timestamp 604a)
(Sixth)
(Second)
208
Module A
101
110
Module
101
Module
101
Module
101
Module
101
Module
101
(Seventh)
701 (w/ Timestamp 604a)
104
Firewall
117

Figure 14

# SET OF SERVERS FOR "MACHINE-TO-MACHINE" COMMUNICATIONS USING PUBLIC KEY INFRASTRUCTURE

## CROSS-REFERENCE TO RELATED APPLICATIONS

**[0001]** The subject matter of this application is related to the subject matter of U.S. patent application Ser. No. 14/023,181, filed Sep. 10, 2013 in the name of John Nix, entitled "Power Management and Security for Wireless Modules in 'Machine-to-Machine' Communications," which is hereby incorporated by reference in its entirety.

**[0002]** The subject matter of this application is also related to the subject matter of U.S. patent application Ser. No. 14/039,401, filed Sep. 27, 2013 in the name of John Nix, entitled "Secure PKI Communications for 'Machine-to-Machine' Modules, including Key Derivation by Modules and Authenticating Public Keys," which is hereby incorporated by reference in its entirety.

**[0003]** The subject matter of this application is also related to the subject matter of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix, entitled "Systems and Methods for 'Machine-to-Machine' (M2M) Communications Between Modules, Servers, and an Application using Public Key Infrastructure (PKI)," which is hereby incorporated by reference in its entirety.

## BACKGROUND

**[0004]** 1. Technical Field

**[0005]** The present methods and systems relate to communications between a set of servers and a plurality of modules, and more particularly, to methods and systems for supporting secure, efficient, and flexible communications using Internet Protocol networks, where a server can communicate with both a "machine-to-machine" module and an application.

**[0006]** 2. Description of Related Art

**[0007]** The combination of "machine-to-machine" (M2M) communications and using low-cost sensors, Internet connections, and processors is a promising and growing field. Among many potential benefits, M2M technologies allow the remote monitoring and/or control of people, assets, or a location where manual monitoring is not economic, or costs can be significantly reduced by using automated monitoring as opposed to manual techniques. Prominent examples today include vending machines, automobiles, alarm systems, and remote sensors. Fast growing markets for M2M applications today include tracking devices for shipping containers or pallets, health applications such as, but not limited to, the remote monitoring of a person's glucose levels or heartbeat, monitoring of industrial equipment deployed in the field, and security systems. Many M2M applications leverage either wired Internet connections or wireless connections, and both types of connections continue to grow rapidly. M2M applications may also be referred to as "the Internet of things".

**[0008]** M2M communications can provide remote control over actuators that may be connected to a M2M device, such as, but not limited to, 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 an M2M device can utilize one or a series of sensor measurements. An M2M device may also be referred to as a "wireless module" or also simply a module. As one example, if a building or room is too cold, then temperature can be reported to a central server by an M2M device and the server can instruct the M2M device to turn on a switch that activates heat or adjusts a thermostat. As the costs for computer and networking hardware continue to decline, together with the growing ease of obtaining either wired or wireless Internet access for small form-factor devices, the number of economically favorable applications for M2M communications grows.

**[0009]** Many M2M applications can leverage wireless networking technologies. Wireless technologies such as, but not limited to, wireless local area networks and wireless wide area networks have proliferated around the world over the past 15 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 $3^{rd}$ Generation Partnership Project's (3GPP) $3^{rd}$ Generation (3G) Universal Mobile Telecommunications System (UMTS) and $4^{th}$ Generation (4G) Long-term Evolution (LTE), LTE Advanced, and the Institute of Electrical and Electronics Engineers' (IEEE) 802.16 standard, also known as WiMax. The use of wireless technologies with "machine-to-machine" communications creates new opportunities for the deployment of M2M modules in locations without fixed-wire Internet access, but also creates a significant new class of problems that need to be solved.

**[0010]** First, many wireless wide-area networking standards were designed and optimized for mobile phones, which may be continuously connected to the network during the day (i.e. non-sleeping hours for most subscribers while they may charge phones at night), in order to receive inbound phone calls and messages. In this case, the radio may be in an idle state but utilizing discontinuous reception, but the radio is still active and drawing power in order to receive and process incoming signaling from the network such as, but not limited to, a Public Land Mobile Network (PLMN). A need exists in the art to make wireless M2M communications efficient in order to conserve battery life and radio-frequency spectrum resources.

**[0011]** Since the packets transmitted and received by a wireless module will likely traverse the public Internet for many applications, a need exists in the art to (i) prevent eavesdropping at intermediate points along the path of packets transmitted and received, (ii) allow endpoints to verify the identity of the source of packets received. A need exists in the art for a wireless module and a monitoring server to leverage established public key infrastructure (PKI) techniques and algorithms. A need exists in the art for communication to be secured without requiring the established, but relatively processing, bandwidth, and energy intensive security protocols, such as, but not limited to, IPSec, Transport Layer Security (TLS), and Secure Socket Layer (SSL) between a module and a server. The establishment of theses links requires extra overhead in the form of packet handshakes and/or key exchanges at levels including the network and transport layer of the traditional Open Systems Interconnection (OSI) model.

**[0012]** M2M applications frequently require small, periodic messages sent between a wireless module and a monitoring server, where the wireless module sleeps between the messages. M2M applications may leverage wired modules as well which can also sleep between messages. During relatively long periods of sleep such as 30 minutes or more, the a

wireless or wired network with intermediate firewalls will often tear down the network and/or transport layer connections, which means the wireless module would need to renegotiate or reestablish the secure tunnels each time the wireless module wakes and seeks to send a relatively small message to a server. A need exists in the art for supporting established security protocols with an external application, without requiring them to be implemented on a module due to the relatively long periods of sleep and other complexities from inactivity in the module.

**[0013]** Next, a need exists in the art for the communication between a module and a monitoring server to be highly energy and bandwidth efficient in order to reduce energy consumption over the operating lifetime of a module. A limiting factor for a wireless module for M2M applications deployed or installed into the field is the lifetime of the battery of the wireless module. If the transmission techniques for the wireless module are not energy efficient, the system will require more frequent manual intervention for the replacement or recharging of batteries. The energy saving techniques for transmitting and receiving data should leverage established Internet protocols. For wired modules operating for years or decades, a significant cost will be the power consumed from land-line power.

**[0014]** Further, a need exists in the art for the secure, energy efficient communications that support Internet protocols to support intermediate firewalls that may exist along the path of packets sent and received by both a wireless module and a monitoring server. Without support for communication through an intermediate firewall, packets may be blocked by the firewall and the M2M application would not properly function in this case. Currently, there are dozens of manufacturers and form-factors of modules, and this diversity will continue to increase for the foreseeable future. By leveraging standards such as the Internet and PKI technologies, an efficient, secure, and highly scalable system of communicating could support the wide variety of modules.

**[0015]** In addition, the utilization of PM technologies in modules can increase security, but a number of technical challenges must be addressed. These challenges increase if a deployed module required updated private/public key pairs after operation begins. The typical paradigm of "swapping out a SIM card" (which also depend on a pre-shared secret key Ki embedded in the card) with mobile phones may not be applicable or cost effective with modules, where swapping out the SIM card could be burdensome. A need exists in the art to allow for a deployed module to securely and automatically begin using new private and public keys (i.e. without human intervention such as swapping out a SIM card). Newer PKI technologies may offer a wide variety of algorithms for ciphering with public keys, and a need exists in the art for the utilization of new public and private keys to support the wide variety of algorithms, even after a module has been installed. A need exists in the art for a scalable and secure method of associating a module identity with a module public key, when the module begins utilizing a new public key. A need exists in the art for a module to efficiently be able to utilize multiple public/private key pairs at the same time, such as with different service providers or different applications simultaneously.

**[0016]** Another desirable feature is for an M2M module to efficiently and securely communicate with applications. Applications can include a web-based interface for users to view status or input settings for a plurality of modules, and the modules may be associated with an M2M service provider. However, a set of PKI algorithms, keys, and communication protocols within used by the module for efficient communications module may not be directly compatible with an application. As one example, the application on a web server may prefer to use a transport layer security (TLS) protocol with transmission control protocol (TCP) datagrams, while for energy efficiency and to conserve battery life, an M2M module may prefer to use user datagram protocol (UDP). A need exists in the art for an intermediate server to securely translate secure communications to/from a module into secure communication from/to an application. As another example, it would be desirable for a module to support elliptic key cryptography (ECC), while the application may support RSA-based cryptography, and therefore a need exists in the art for a server to securely translate between the two cryptographic methods, thereby allowing the M2M module to communicate with the application.

**[0017]** And other needs exist in the art as well, as the list recited above is not meant to be exhaustive but rather illustrative.

## SUMMARY

**[0018]** Methods and systems are provided for secure and efficient communication using a server to communicate with modules and an application. The modules and application can support "Machine to Machine" communications. The methods and systems contemplated herein can also support other applications as well, including mobile phone handsets connecting to a wireless network. An objective of the invention is to address the challenges noted above for securing the deployment of modules that utilize PKI algorithms and keys, as well as increasing efficiency in order to reduce power consumption, including extending the battery life of a module, if present. More efficient communication can also conserve valuable radio-frequency spectrum, among other benefits. Using a server for secure and reliable communication of data between an application and a module can increase the value and usefulness of modules for a user.

**[0019]** An exemplary embodiment may take the form of methods and systems for a server to securely receive data from a module and forward the data to an application server, and an application may operate on the application server. The application can include a graphical user interface for a user to visually see reports and/or control modules. The module, server, and application can preferably include a set of cryptographic algorithms for use in sending and receiving data. The cryptographic algorithms can include asymmetric ciphering algorithms, symmetric ciphering algorithms, secure hash algorithms, digital signature algorithms, key pair generation algorithms, a key derivation function, and/or a random number generator.

**[0020]** The module can utilize the set of cryptographic algorithms to securely generate or derive a module private key and a module public key. The module private key and module public key can be generated either (i) upon initial use or installation of the module, or (ii) at a subsequent time after initial use such as when a new set of key pairs are required or are useful for continued operation of the module. After deriving the module public key and module private key, the module private key is preferably recorded in a secure or protected location in a nonvolatile memory within the module. In one embodiment, the module may then utilize a recorded pre-shared secret key to authenticate with a server that also

records or has access to the pre-shared secret key and the module identity. The authentication could comprise either using message digest with the pre-shared secret key, or using the pre-shared secret key in processing a symmetric ciphering key, and the authentication may also utilize a second key derived by both the module and the server using the pre-shared secret key. After authentication, the server can authoritatively record the derived module public key with the module identity in a database. Thus, the use of a pre-shared secret key can ensure the submitted module public key is validly associated with the module and module identity.

**[0021]** The server can (i) include a private key associated with the server, and (ii) receive the derived module public key. The server public key can leverage established public key infrastructure (PKI) standards, such as, but not limited to, X.509 v3 certificates and RSA or elliptic curve cryptography (ECC) algorithms and include a digital signature from a certificate authority. The server can use a module controller and an operating system plus a connection to the Internet to monitor a socket for incoming messages from a module. After receiving the module public key, including potentially after a period of sleep or dormancy by the module, the server can receive a message, where the message includes a module identity and a module encrypted data. The module encrypted data can include a server instruction, a security token, and additional data such as, but not limited to, a sensor measurement. The server can decrypt the module encrypted data using the received module public key and extract plaintext data from the module encrypted data.

**[0022]** The server can establish a secure connection with the application server using a secure connection setup, which could comprise the initial handshake messages for a transport-layer security protocol such as, but not limited to, transport layer security (TLS) or IPSec. The secure connection setup can include the transfer of a server public key and an application server public key. The server can send an application message to the application server using a secure connection data transfer, where the application message includes data received from the module such as, but not limited to, a sensor measurement or sensor data. The server can use (i) an RSA-based asymmetric ciphering algorithm and first server public key with the application server to securely transfer a first symmetric key to the application server, and (ii) an ECC-based asymmetric ciphering algorithm and second server public key with the module to securely transfer a second symmetric key to the module. In an exemplary embodiment the server may also preferably use a transmission control protocol (TCP) with the application server and a user datagram protocol (UDP) with the module. The application message to the application server can include a server identity, an encrypted update instruction, and the sensor data. The sensor data may also include a sensor identity. The server can use a first Internet protocol address and port (IP:port) number for receiving the message from the module and a second IP:port number for sending the application message to the application server. The application server can record the sensor data in an application database for subsequent processing and analysis for a user or other business or commercial needs.

**[0023]** In another embodiment, the module may be deployed within a wireless network such as, but not limited to, a 4G LTE network or a WiFi network, and the module may comprise a wireless module. The module can change state between a sleep state and an active state, wherein the sleep state may utilize a few milliwatts or less and the active state, including transmission of radio signals, may utilize several hundred milliwatts of power or more. After being installed next to a monitored unit, the wireless module can wake from a sleep or dormant state, utilize a sensor to collect data associated with the monitored unit, connect to the wireless network and the Internet, and send the sensor data to a server. During an active period, the module can use a UDP IP:port number to both send a message to the server and receive a response to the server. The message as a UDP datagram can be a UDP Lite datagram and with a checksum only applied to the packet header. A UDP Lite datagram with sensor data can include channel coding for the body of the datagram to mitigate the effect of bit errors. Or, a regular UDP packet could be sent in multiple copies in order to provide forward error correction.

**[0024]** In another embodiment of the present invention, the application server may send an application message to the server using a secure connection data transfer. The application message could be encrypted using a first server public key and could include a module identity and a module instruction. The module instruction can include an actuator setting, and also optionally an actuator identity (since the module may include multiple actuators). The server can decrypt encrypted data within the application message and record the module identity and module instruction in memory or a module database. Since the module can transition between periods of sleep and active states to conserve power, after receiving the application message the server can wait until a next message is received from the module with the module identity before sending the module instruction in a response. After waiting for the next message, the server can send the module instruction to the module in a server encrypted data using a second server public key. The first and second server public keys can use different cryptographic algorithms that are not directly compatible (i.e. the first server public key could be RSA-based and the second server public key could be ECC-based).

**[0025]** In another embodiment, the server can securely send the module a set of cryptographic parameters, where the set of cryptographic parameters includes values to define an equation for an elliptic curve. The values could comprise constants and variables such that the module can calculate a new elliptic curve, and the elliptic curve can be different than standard, published curves. The set of cryptographic parameters could be sent from the server to the module in a server encrypted data, where the server encrypted data was processed using any of (i) a first module public key, (ii) a symmetric key, and (iii) a shared secret key. The module can use the set of cryptographic parameters, a random number generator, and a key generation function within a cryptographic algorithms in order to generate a new key pair, which could comprise a second module public key and a second module private key. The module can securely and/or authoritatively send the second module public key to the server, where the security includes the use of the first module public key and/or the shared secret key.

**[0026]** Continuing with this embodiment, after the server confirms the proper receipt of the second, derived module public key in a response message, the server and the module can begin secure communications between them using the second module public key. By using this exemplary embodiment, security can be further increased with the server and module using an elliptic curve that can be unique, non-standard, or defined between them and security therefore

increased. In this exemplary embodiment, the parameters to define the elliptic curve equation are sent securely to the module, so an observer along the flow of data could not observe the elliptic equation being used with a public key.

**[0027]** In yet another embodiment, the server can receive a first message with a module identity and a module encrypted data, where the first module encrypted data includes a first sensor measurement. The server can use a first module public key associated with a first module public key identity to decrypt the first module encrypted data. As one example, (a) the first module encrypted data could be ciphered with a symmetric key, and (b) the symmetric key could have been communicated using the first module public key (including using the first module public key to verify a module digital signature in a session or flow of packets where the symmetric key was transferred), and therefore (c) the module encrypted data could be encrypted using the first module public key. The server can also use a first server public key to decrypt the first module encrypted data, such as, but not limited to, the symmetric key being derived using both the first module public key and the first server public key and a key derivation function within a cryptographic algorithms. The server can extract the first sensor measurement and send the data to an application server in an application message. The application message could be encrypted using a second server public key. The first and second server public keys can be different because they could each be associated with a different algorithm or defining equation.

**[0028]** Continuing with this embodiment, the server can send a module instruction and a set of cryptographic parameters to the module, where the module is instructed to derive a new set of keys, and the module can subsequently derive a second module public key and a second module private key after receiving the module instruction. The module can then send the second module public key, a second module public key identity, and the module identity to the server. The server can receive a second module encrypted data that includes a second sensor data, where the second sensor data is encrypted using the second module public key. As one example, (a) the second module encrypted data could be ciphered with a symmetric key, and (b) the symmetric key could have been communicated using the second module public key (including using the second module public key to verify a module digital signature in a session where the symmetric key was transferred), and therefore (c) the module encrypted data could be encrypted using the second module public key. The server can extract the second sensor data using the second module public key. The server can use the second server public key to send a second application message with the second sensor data to the application server. Note that the module public key can change, but both (i) the second server public key used with the application server and also (ii) keys associated with the application server did not change. In this manner according to this embodiment, a module can derive a new public and private key while a server and application server can continue to communicate using existing public and private keys.

**[0029]** In another embodiment, a system supporting M2M communications can include a set of application servers, a set of servers, and a set of modules. The set of servers can record and query data from a shared module database. At least one of the application servers can process or originate a module instruction, and send the module instruction with a module identity to the shared module database. A module with the module identity may wake from a dormant state and send a message with a module identity and a module encrypted data to a server, where the server was a member of the set of servers. Upon receiving the message and verifying the message originated from a module with the module identity, the server can poll the shared module database using the module identity. The shared module database can return the module instruction that was recorded by the application server. The server can send the module instruction to the module with the module identity in a response. Upon executing the module instruction, the module can send a confirmation with a timestamp to the server in a module encrypted data. The server can then send the timestamp and a module identity in an application message to the application server, and in this manner the application server can determine a time when the module instruction was processed by the module.

**[0030]** In an exemplary embodiment, a module with a module identity can derive its own public and private keys after distribution of the module using a set of cryptographic parameters. A set of servers can receive a message that uses a module identity, where the module identity can be verified using at least one of a module digital signature and a shared secret key. The set of servers can send the module with the module identity the set of cryptographic parameters. Over time, the module can use at least a subset of the cryptographic parameters to derive multiple pairs of module public and private keys. Over time, the server can receive a series of module public keys with the module identity and use a previous module public key in the series to verify and/or authenticate a message with a module public key.

**[0031]** 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

**[0032]** Various exemplary embodiments are described herein with reference to the following drawings, wherein like numerals denote like entities.

**[0033]** FIG. **1***a* is a graphical illustration of an exemplary system, where a server and a module connect to the Internet, in accordance with exemplary embodiments;

**[0034]** FIG. **1***b* is a graphical illustration of hardware, firmware, and software components for a module, in accordance with exemplary embodiments;

**[0035]** FIG. **1***c* is a graphical illustration of hardware, firmware, and software components for a server, in accordance with exemplary embodiments;

**[0036]** FIG. **1** *d* is a graphical illustration of hardware, firmware, and software components for an application server, in accordance with exemplary embodiments;

**[0037]** FIG. **1***e* is a graphical illustration of components within a module, in accordance with exemplary embodiments;

**[0038]** FIG. **1***f* is a graphical illustration of components within a server, in accordance with exemplary embodiments;

**[0039]** FIG. **1** *g* is a graphical illustration of components in a set of cryptographic algorithms, in accordance with exemplary embodiments;

**[0040]** FIG. **1***h* is a graphical illustration of an exemplary system that includes a user, an application, a set of servers, and a set of modules, in accordance with exemplary embodiments;

**[0041]** FIG. **2** is a graphical illustration of an exemplary system, where a module sends a message to a server, and where the server responds to the message, in accordance with exemplary embodiments;

**[0042]** FIG. **3** is a flow chart illustrating exemplary steps for a server to receive a message from a module, in accordance with exemplary embodiments;

**[0043]** FIG. **4** a is a flow chart illustrating exemplary steps for a server to process a message, including verifying a module's identity and decrypting data, in accordance with exemplary embodiments;

**[0044]** FIG. **5***a* is a flow chart illustrating exemplary steps for a server to process a response for a module, including sending and signing a module instruction, in accordance with exemplary embodiments;

**[0045]** FIG. **5***b* is a flow chart illustrating exemplary steps for a server to communicate with a module that has derived a public key and private key, in accordance with exemplary embodiments;

**[0046]** FIG. **6***a* is a simplified message flow diagram illustrating an exemplary message received by a server, and an exemplary response sent from the server, in accordance with exemplary embodiments;

**[0047]** FIG. **6***b* is a simplified message flow diagram illustrating an exemplary message received by a server, wherein the message includes a derived module public key, in accordance with exemplary embodiments;

**[0048]** FIG. **7** is a simplified message flow diagram illustrating an exemplary system with exemplary data transferred between a module and an application using a server, in accordance with exemplary embodiments;

**[0049]** FIG. **8** is a simplified message flow diagram illustrating an exemplary system with exemplary data transferred between a module and an application using a server, in accordance with exemplary embodiments;

**[0050]** FIG. **9** is a simplified message flow diagram illustrating exemplary data transferred between (i) a server and an application and between (ii) a server and a module, in accordance with exemplary embodiments;

**[0051]** FIG. **10** is a flow chart illustrating exemplary steps for a set of servers to communicate with a module, in accordance with exemplary embodiments;

**[0052]** FIG. **11** is a flow chart illustrating exemplary steps for a set of servers to communicate with a module and an application server, in accordance with exemplary embodiments;

**[0053]** FIG. **12** is a simplified message flow diagram illustrating an exemplary system with exemplary data transferred between a module and an application using a server, in accordance with exemplary embodiments;

**[0054]** FIG. **13** is a simplified message flow diagram illustrating an exemplary system with exemplary data transferred between a module and an application using a server, in accordance with exemplary embodiments;

**[0055]** FIG. **14** is a graphical illustration of an exemplary system that includes a set of application servers, a set of servers, and a set of modules, in accordance with exemplary embodiments.

## DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS OF THE INVENTION

**[0056]** FIG. **1***a*

**[0057]** FIG. **1***a* is a graphical illustration of an exemplary system, where a server and a module connect to the Internet, in accordance with exemplary embodiments. The system **100** includes a module **101** operating within a wireless network **102**. System **100** can also include a module provider **109**, an Internet **107**, and an M2M service provider **108**, a certificate authority **118**, and a monitored unit **119**. M2M service provider **108** can include a server **105**. System **100** is illustrated without specific packet transmissions between module **101** and M2M service provider **108**. Examples of the communications and messages pertaining to the present invention will be illustrated in later Figures. As contemplated herein, machine-to-machine communications may comprise communication between a module **101** and a server **105**, such that data can be transferred between the two with minimal manual intervention, although manual intervention can be required to set up system **100** and any occasional manual maintenance required. As contemplated herein, machine-to-machine communications may also be referred to as "the Internet of things" (IoT). Also note that module **101** may comprise a wireless module, such that module **101** can communicate with wireless network **102** using a radio and an antenna. A wireless or a wired configuration for module **101** can be utilized in the present invention.

**[0058]** If module **101** operates as a wireless module, module **101** and wireless network **102** can communicate using a base station **103**. Module **101** and wireless network **102** can utilize a variety of wireless technologies to communicate, including WiFi, WiMax, a 2nd generation wireless wide area network (WAN) technology such as, but not limited to, General Packet Radio Services (GPRS) or Enhanced Data rates for GSM Evolution (EDGE), 3rd Generation Partnership Project (3GPP) technology such as, but not limited to, 3G, 4G LTE, or 4G LTE Advanced, and other examples exist as well. A wired module **101** can connect to the Internet **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).

**[0059]** Generally, the communication techniques described herein can be independent of the network technologies utilized at the physical and data-link layers, so long as the underlying network provides access to the Internet **107** and supports Internet Protocols (IP). The Internet **107** can be an IPv4 or an IPv6 packet-switched based network that utilizes standards derived from the Internet Engineering Task Force, such as, but not limited to, RFC 786 (User Datagram Protocol), RFC 793 (Transmission Control Protocol), and related protocols. The Internet **107** can be the public Internet comprising globally routable IP addresses, or a private network that utilizes private IP addresses. Although Internet **107** is illustrated as the globally routable public Internet in FIG. **1**, Internet **107** could also be a private Internet that is (i) not globally routable and (ii) only accessible to authorized modules and servers. As one example of a private Internet **107**, Internet **107** could use private IP addresses for nodes on the network, and in this case Internet **107** could be referred to as an intranet or private network. Alternatively, Internet **107** could be a private network layered on top of the publicly routable Internet via secured and encrypted connections. The specific numbers for IP addresses and port numbers shown in FIG. **1** and other figures are illustrative and any valid IP address or port number can be used, including an IPv4 and an IPv6 address.

**[0060]** When operating in a wireless network configuration, module **101** can access the Internet **107** via the wireless network **102**. In the wireless network configuration, module

**101** can be a wireless handset, a cellular phone, a smartphone, a tablet computer, a laptop, a computer with a radio, a tracking device, or a circuit board with a radio that accesses wireless network **102**. Examples of wireless modules that utilize a wireless WAN such as, but not limited to, 2G and 3G networking technologies include the Motorola® G24-1 and Huawei® MC323. Example manufacturers of wireless modules in 2012 include Sierra Wireless® and Telit®. In a wired configuration (not shown), module **101** can be a computer, security camera, security monitoring device, networked controller, etc. A more detailed depiction of exemplary components of a module **101** is included in FIG. **1***b* and FIG. **1***e* below. Module **101** could also comprise a "point of presence" payment terminal, such that a sensor **101***f* associated with module **101** could collect payment information such as, but not limited to, an account number from a credit card or similar payment card. Module **101** could communicate with the payment card via a magnetic reader or near-field wireless communications, and in this case the magnetic reader or antenna for near-field communications can function as a sensor. Module **101** could also operate as a "smartcard" such that an end user presents module **101** to merchants for payments.

**[0061]** Wireless network **102** may comprise either a wireless local area network (LAN) such as, but not limited to, an 802.11 WLAN, Bluetooth, or Zigbee among other possibilities, and module **101** operating in wireless mode could communicate with a base station **103** of a wireless network **102** using a radio and an antenna. Wireless network **102** could operate as a Mode II device according to FCC Memorandum Opinion and Order (FC-12-36) and related white space regulation documents. If module **101** supports IEEE 802.15.4, then wireless network **102** could be a Zigbee network, an ISA100.11a standards-based network, or a 6LoWPAN network as described by IETF RFC 4944. Other possibilities exist as well for the wireless technology utilized by a wireless network **102** and module **101**, operating in a wireless mode, without departing from the scope of the present invention.

**[0062]** Module **101** can collect data regarding a monitored unit **119** and periodically report status to an M2M service provider **108** or a server **105**. Examples of a monitored unit **119** can include a vending machine, an alarm system, an automobile or truck, a standard 40-foot or 20-foot shipping container, or industrial equipment such as, but not limited to, a transformer on an electrical grid or elevator in a building. Additional examples of a monitored unit **119** include can also include a pallet for shipping or receiving goods, an individual box of pharmaceuticals, a health monitoring device attached to a person such as, but not limited to, a pacemaker or glucose monitor, and a gate or door for opening and closing. Other examples exist as well without departing from the scope of the present invention. Module **101** can utilize a sensor to measure and collect data regarding a parameter of monitored unit **119** such as, but not limited to, temperature, physical location potentially including geographical coordinates from a Global Positioning System (GPS) receiver, radiation, humidity, surrounding light levels, surrounding RF signals, weight, vibration and/or shock, voltage, current, and/or similar measurements.

**[0063]** As illustrated in FIG. **1***a*, wireless network **102** may include a wireless network firewall **104** and M2M service provider **108** may include a server network firewall **124**. These firewalls may be used to secure communication at the data link, network, transport, and/or application layers of communications using the Internet **107**. Firewalls **104** and **124** could perform network address translation (NAT) routing or operate as symmetric firewalls, and selectively filter packets received through Internet **107** in order to secure system **100**. The firewall functionality of firewalls **104** and **124** could be of many possible types, including a symmetric firewall, a network-layer firewall that filters inbound packets according to pre-determined rules, an application-layer firewall, or a NAT router, as examples. Although a single firewall **104** and **124** is illustrated in wireless network **102** (or a wired network **102** or simply "network **102**") and with M2M service provider **108**, respectively, firewall **104** and **124** may each comprise multiple firewalls that operate in conjunction and the combined operation may be considered a single firewall **104** and **124**, respectively. Firewall **104** and/or firewall **124** can include a firewall port binding timeout value **117** (illustrated in FIG. **2**), which can represent the time allowed for an inbound packet from the Internet **107** to pass through firewall **104** or firewall **124** after module **101** or server **105**, respectively, sends a packet out. Firewall port binding timeout value **117** may be determined on a per-protocol basis, such as an exemplary time of 60 seconds for UDP packets and 8 minutes for TCP packets, although other time values for a firewall port binding timeout value **117** are possible as well. Inbound packets from Internet **107** to module **101** may be dropped by firewall **104** after a time exceeding firewall port binding timeout value **117** has transpired since the last packet transmitted by module **101**.

**[0064]** According to a preferred exemplary embodiment, module **101** may preferably record a module private key **112**. As described in additional figures below, module **112** can generate a key pair comprising a module private key **112** and a module public key **111**, where module private key **112** resides within module **101** and may not be shared or transmitted to other parties. Alternatively, the present invention also contemplates that module **101** does not derive its own module private key **112**, and rather module private key **112** is securely loaded or transmitted to module **101**. Module **101** may also be associated with a module provider **109**. Module provider **109** could be a manufacturer or distributor of module **101**, or may also be the company that installs and services module **101** or associates module **101** with monitored unit **119**. Module provider **109** can record a module public key **111** and a certificate **122** (illustrated below in FIG. **1** *e*) for module **101**. Module public key **111** may be associated with a module public key identity **111***a*, which could be an identifier of module public key **111**.

**[0065]** In embodiments, a module **101** may utilize multiple module public keys **111** over the lifetime of module **101** (including multiple corresponding module private keys **112**), and module public key identity **111***a* can be used to select and/or identify the correct module public key **111**. Module public key identity **111***a* could be a string or sequence number uniquely associated with module public key **111** for a given module **101** (i.e. module public key identity **111***a* does not need to be globally unique). As illustrated in FIG. **1***a*, module public key identity **111***a* may preferably not be included in the string or number comprising module public key **111**, but rather associated with the string or number comprising module public key **111**, and in this case the two together (module public key identity **111***a* and the string or number for module public key **111**) can refer to module public key **111** as contemplated herein.

**[0066]** The module public key **111** can optionally be signed by a certificate authority **118** in order to confirm the identity

of module **101** and/or the identity of module provider **109**. Module provider **109** can also function as a certificate authority **118** for module **101**. Thus, the validity of module public key **111**, possibly recorded in a certificate **122** (illustrated in FIG. **1***e*) could be checked with module provider **109**, and the wireless module provider's **109** provider public key **120** could be checked against certificate authority **118**. Other configurations for signing public keys and using certificates with public keys are possible as well without departing from the scope of the present invention.

**[0067]** Public keys and private keys as contemplated in the present invention, including module public key **111** and module private key **112** and additional keys described herein, may leverage established standards for Public Key Infrastructure (PKI). Public keys may be formatted according to the X.509 series of standards, such as, but not limited to, X.509 v3 certificates, and subsequent or future versions, and these keys may be considered cryptographic keys. The keys can support standards such as, but not limited to, the International Organization for Standardization (ISO) ISO/IEC 9594 series of standards (herein incorporated by reference) and the Internet Engineering Task Force (IETF) RFC 5280 titled "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" (herein incorporated by reference), including future updates to these standards.

**[0068]** Module public key **111** and module private key **112**, as well as the other private and public keys described within the present invention, could be generated using standard software tools such as, but not limited to, Openssl, and other tools to generate public and private keys exist as well. Public and private keys as contemplated herein could be recorded in a file such as, but not limited to, a *.pem file (Privacy-enhanced Electronic Mail), a file formatted according to Basic Encoding Rules (BER), Canonical Encoding Rules (CER), or Distinguished Encoding Rules (DER), or as text or binary file. Other formats for public and private keys may be utilized as well, including proprietary formats, without departing from the scope of the present invention. As contemplated herein, a key may also comprise either a public key or a private key. A public key as contemplated herein may also be considered a certificate or a public certificate. A private key as contemplated herein may also be considered a secret key.

**[0069]** Other configurations besides the one illustrated in FIG. **1***a* are possible as well. Server **105** could reside within wireless network **102** in a data center managed by wireless network **102**. Wireless network **102** could also operate as a module provider **109**. Although a single module **101** and server **105** are illustrated in FIG. **1***a*, system **100** could comprise a plurality of each of these elements. Module **101** could also record sensor data pertaining to a plurality of monitored units **119**. Module **101** could be mobile, such as physically attached to a truck or a pallet, and module **101** could connect to a series of different wireless networks **102** or base stations **103** as module **101** moves geographically. Other configurations are possible as well without departing from the scope of the present invention.

**[0070]** FIG. **1***b*

**[0071]** FIG. **1***b* is a graphical illustration of hardware, firmware, and software components for a module, in accordance with exemplary embodiments. FIG. **1***b* is illustrated to include many components that can be common within a module **101**, and module **101** may also operate in a wireless configuration in order to connect with a wireless network **102**. Module **101** may consist of multiple components in order to collect sensor data or control an actuator associated with a monitored unit **119**. In a wireless configuration, the physical interface **101***a* of module **101** may support radio-frequency (RF) communications with networks including a wireless network **102** via standards such as, but not limited to, GSM, UMTS, mobile WiMax, CDMA, LTE, LTE Advanced, and/or other mobile-network technologies. In a wireless configuration, the physical interface **101***a* may also provide connectivity to local networks such as, but not limited to, 802.11 WLAN, Bluetooth, or Zigbee among other possibilities. In a wireless configuration, module **101** could use a physical interface **101***a* be connected with both a wireless WAN and wireless LAN simultaneously. In a wired configuration, the physical interface **101***a* can provide connectivity to a wired network such as, but not limited to, through an Ethernet connection or USB connection.

**[0072]** The physical interface **101***a* can include associated hardware to provide the connections such as, but not limited to, radio-frequency (RF) chipsets, a power amplifier, an antenna, cable connectors, etc., and additional exemplary details regarding these components are described below in FIG. **1***e*. Device driver **101***g* can communicate with the physical interfaces **101***a*, providing hardware access to higher-level functions on module **101**. Device drivers **101***g* may also be embedded into hardware or combined with the physical interfaces. Module **101** may preferably include an operating system **101***h* to manage device drivers **101***g* and hardware resources within module **101**. The operating systems described herein can also manage other resources such as, but not limited to, memory and may support multiple software programs operating on module **101** or server **105**, respectively, at the same time. The operating system **101***h* can include Internet protocol stacks such as, but not limited to, a User Datagram Protocol (UDP) stack, Transmission Control Protocol (TCP) stack, a domain name system (DNS) stack, etc., and the operating system **101***h* may include timers and schedulers for managing the access of software to hardware resources. The operating system shown of **101***h* can be appropriate for a low-power device with limited memory and CPU resources (compared to a server **105**). An example operating system **101***h* for module **101** includes Linux, Android® from Google®, Windows® Mobile, or Open AT® from Sierra Wireless®. Additional example operating systems **101***h* for module **101** include eCos, uC/OS, LiteOs, and Contiki, and other possibilities exist as well without departing from the scope of the present invention.

**[0073]** A module program **101***i* 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, but not limited to, remote monitoring of sensors and remote activation of actuators. Module program **101***i* could also be a software routine, subroutine, linked library, or software module, according to one preferred embodiment. As contemplated herein, a module program **101***i* may be an application operating within a smartphone, such as, but not limited to, an iPhone® or Android®-based smartphone, and in this case module **101** could comprise the smartphone. The application functioning as a module program **101***i* could be downloaded from an "app store" associated with the smartphone. Module program **101***i* can include data reporting steps **101***x*, which can provide the functionality or CPU **101***b* instructions for collecting sensor data, sending messages to server **105**, and receiving responses from server **105**, as described in the present invention.

**[0074]** Many of the logical steps for operation of module **101** can be performed in software and hardware by various combinations of sensor **101***f*, actuator **101***y*, physical interface **101***a*, device driver **101***g*, operating system **101***h*, module program **101***i*, and data reporting steps **101***x*. When module **101** is described herein as performing various actions such as acquiring an IP address, connecting to the wireless network, monitoring a port, transmitting a packet, sending a message, receiving a response, or encrypting or signing data, specifying herein that module **101** performs an action can refer to software, hardware, and/or firmware operating within module **101** illustrated in FIG. **1***b* performing the action. Note that module **101** may also optionally include user interface **101***j* 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 modules such as, but not limited to, a few LED lights or LCD display, and thus user interfaces are not described in detail here. User interface **101***j* could comprise a touch screen if module **101** operates as a smartphone or mobile phone. As illustrated in FIG. **1***b*, module **101** can optionally omit a user interface **101***j*, since no user input may be required for many M2M applications, although a user interface **101***j* could be included with module **101**.

**[0075]** Module **101** may be a computing device that includes computer components for the purposes of collecting data from a sensor **101***f* or triggering an action by an actuator **101***y*. Module **101** may include a central processing unit (CPU) **101***b*, a random access memory (RAM) **101***e*, and a system bus **101***d* that couples various system components including the random access memory **101***e* to the processing unit **101***b*. The system bus **101***d* 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. Note that the computer components illustrated for the module **101** in FIG. **1***b* may be selected in order to minimize power consumption and thereby maximize battery life, if module **101** includes a battery and is not attached to external power. In addition, the computer components illustrated for the module **101** in FIG. **1***b* may also be selected in order to optimize the system for both long periods of sleep relative to active communications and also may be optimized for predominantly uplink (i.e. device to network) communications with small packets or messages. The computer components illustrated for the module **101** in FIG. **1***b* may also be general-purpose computing components, and specialized components are not required in order to utilize many of the embodiments contemplated herein.

**[0076]** Module **101** may include a read-only memory (ROM) **101***c* which can contain a boot loader program. Although ROM **101***c* is illustrated as "read-only memory", ROM **101***c* could comprise long-term memory storage chipsets or physical units that are designed for writing once and reading many times. As contemplated within the present invention, a read-only address could comprise a ROM **101***c* memory address or another hardware address for read-only operations accessible via bus **101***d*. Changing data recorded in a ROM **101***c* can require a technician have physical access to module **101**, such as, but not limited to, removing a cover or part of an enclosure, where the technician can subsequently connect equipment to a circuit board in module **101**, including replacing ROM **101***c*. ROM **101***c* could also comprise a nonvolatile memory, such that data is stored within ROM **101***c* even if no electrical power is provided to ROM **101***c*. Although not illustrated in FIG. **1***b*, but illustrated in FIG. **1***e* below, module **101** could also include a flash memory **101***w*. Module program **101***i*, data reporting steps **101***x*, operating system **101***h*, or device driver **101***g* could be stored in flash memory **101***w* within module **101** when the module is powered off. These components and/or instructions could be moved from a flash memory **101***w* (not shown in FIG. **1***b* but shown in FIG. **1***e*) into RAM **101***e* when the module is powered on. Note that ROM **101***c* could be optionally omitted or included in a memory unit within CPU **101***b* (not shown).

**[0077]** Although the exemplary environment described herein employs ROM **101***c* and RAM **101***e*, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a module **101**, such as, but not limited to, memory cards, subscriber identity module (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 invention. The memory and associated hardware illustrated in FIG. **1***b* provide nonvolatile storage of computer-executable instructions, data structures, program modules, module program **101***i*, and other data for computer or module **101**. Note the module **101** may include a physical data connection at the physical interface **101***a* such as, but not limited to, a miniaturized universal serial bus adapter, firewire, optical, or other another port and the computer executable instructions such as, but not limited to, module program **101***i*, data reporting steps **101***x*, operating system **101***h*, or device driver **101***g* can be initially loaded into memory such as, but not limited to, ROM **101***c* or RAM **101***e* through the physical interface **101***a* before module **101** is given to an end user, shipped by a manufacturer to a distribution channel, or installed by a technician. In addition, the computer executable instructions such as, but not limited to, module program **101***i*, data reporting steps **101***x*, operating system **101***h* or device driver **101***g* could be transferred wirelessly to module **101**. In either case (wired or wireless transfer of computer executable instructions), the computer executable instructions such as module program **101***i*, data reporting steps **101***x*, operating system **101***h*, or device driver **101***g* could be stored remotely on a disk drive, solid state drive, or optical disk (external drives not shown).

**[0078]** A number of program modules may be stored in RAM **101***e*, ROM **101***c*, or possibly within CPU **101***b*, including an operating system **101***h*, device driver **101***g*, an http client (not shown), a DNS client, and related software. Further, the module program **101***i* and/or data reporting steps **101***x* can perform the various actions described in the present invention for the module **101** through instructions the module program **101***i* and/or data reporting steps **101***x* provide to the CPU **101***b*. A user may enter commands and information into module **101** through an optional user interface **101***j*, 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 **101***j* illustrated in FIG. **1***b* can also comprise the description of a user interface **101***j* within U.S. patent application Ser. No. 14/039,401, filed Sep. 27, 2013 in the name of John Nix, which is herein incorporated in its entirety.

**[0079]** The module **101**, comprising a computer, may operate in a networked environment using logical connections to one or more remote computers, such as the server **105** illustrated in FIG. **1***a*. Server **105** can also function as a general purpose server to provide files, programs, disk storage,

remote memory, and other resources to module **101** usually through a networked connection. Additional details regarding server **105** are provided in FIG. **1***c* below. Additional remote computers with which module **101** communicates may include another module **101** 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 base station **103**, or other common network node. It will be appreciated that the network connections shown throughout the present invention 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.

**[0080]** The module program **101***i* and data reporting steps **101***x* operating within module **101** illustrated in FIG. **1***b* can provide computer executable instructions to hardware such as CPU **101***b* through a system bus **101***d* in order for a module **101** to (i) collect data from a sensor, (ii) change the state of an actuator **101***y*, and (iii) send or receive packets with a server **105**, thus allowing server **105** to remotely monitor or control a monitored unit **119**. The module program **101***i* and/or data reporting steps **101***x* can enable the module **101** to transmit or send data from sensor **101***f* or module **101** by recording data in memory such as RAM **101***e*, where the data can include sensor data, a destination IP:port number, a packet or packet header value, an encryption or ciphering algorithm and key, a digital signature algorithm and key, etc. The data recorded in RAM **101***e* can be subsequently read by the operating system **101***h* or the device driver **101***g*. The operating system **101***h* or the device driver **101***g* can write the data to a physical interface **101***a* using a system bus **101***d* in order to use a physical interface **101***a* to send data to a server **105** using the Internet **107**. Alternatively, the module program **101***i* and/or data reporting steps **101***x* can write the data directly to the physical interface **101***a* using the system bus **101***d*.

**[0081]** The module program **101***i* and/or data reporting steps **101***x*, or operating system **101***h* can include steps to process the data recorded in memory such as, but not limited to, encrypting data, selecting a destination address, or encoding sensor data acquired by (i) a sensor **101***f* or (ii) through a physical interface **101***a* such as, but not limited to, a thermocouple, shock or vibration sensor, light sensor, or global positioning system (GPS) receiver, etc. The module **101** can use the physical interface **101***a* such as, but not limited to, a radio to transmit or send the data from a sensor to a base station **103**. For those skilled in the art, other steps are possible as well for a module program **101***i* or operating system **101***h* to collect data from a sensor **101***f* and send the data in a packet without departing from the scope of the present invention.

**[0082]** Conversely, in order for module **101** to receive a packet or response from server **105**, the physical interface **101***a* can use a radio to receive data from a base station **103**. The received data can include information from a server **105** and may comprise a datagram, a source IP:port number, a packet or header value, an instruction for module **101**, an acknowledgement to a packet that module **101** sent, a digital signature, and/or encrypted data. The operating system **101***h* or device driver **101***g* can use a system bus **101***d* and CPU **101***b* to record the received data in memory such as RAM **101***e*, and the module program **101***i* or operating system **101***h* may access the memory in order to process the received data and determine the next step for the module **101** after receiving the data. The steps within this paragraph may also describe the steps a module program **101***i* or data reporting steps **101***x* can perform in order to receive a packet or a response **209** below. For those skilled in the art, other steps are possible as well for a module program **101***i*, data reporting steps **101***x*, or module **101** to receive a packet or response from a server **105** within the scope of the present invention.

**[0083]** Moreover, those skilled in the art will appreciate that the present invention 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, and the like. The invention 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 modules may be located in both local and remote memory storage devices. In addition, the terms "mobile node", "mobile station", "mobile device", "M2M module", "M2M device", "networked sensor", or "industrial controller" can be used to refer to module **101** or its functional capabilities of (i) collecting sensor data regarding a monitored unit **119**, (ii) changing state of an actuator **101***y* associated with monitored unit **119**, and/or (iii) communicating the data associated with a monitored unit **119** with a wireless network **102**. The function of module **101** and sensor **101***f* could be integrated, and in this case module **101** could also be referred to as a "sensor", "intelligent sensor", or "networked sensor". Further, the term "module" or "monitoring device" can be used to refer to the module program **101***i* when module program **101***i* provides functional capabilities such as reporting data from a sensor **101***f* to a server **105** or receiving instructions for an actuator **101***y* from a server **105**. Other possibilities exist as well for the configuration or combination of components illustrated in FIG. **1***b* without departing from the scope of the present invention.

**[0084]** FIG. **1***c*

**[0085]** FIG. **1***c* is a graphical illustration of hardware, firmware, and software components for a server, in accordance with exemplary embodiments. The illustrated components for the server **105** in FIG. **1***c* include a central processing unit (CPU) **105***b*, a random access memory (RAM) **105***e*, a system bus **105***d*, storage **105***m*, an operating system **105***h*, and a module controller **105***x*. These elements can provide functions equivalent to the central processing unit (CPU) **101***b*, RAM **101***e*, system bus **101***d*, flash memory **101***w*, and an operating system **101***h* described above in FIG. **1***b*, respectively. In general, a server **105** can have higher-end components such as, but not limited to, a larger CPU **105***b* and greater RAM **105***e* in order to support communications with a plurality of modules **101**. Server **105** can comprise a general purpose computer such as, but not limited to, a rack mounted server within a data center or rack, or could also comprise a desktop computer or laptop. Server **105** could also be a specialized computer, with hardware and software selected for supporting a plurality of modules **101** connecting and communicating simultaneously. Operating system **101***h* can comprise an operating system appropriate for a server such as, but not limited to, Linux, Solaris®, or Windows® Server. Server **105** can preferably include at least one wired Ethernet connection with high bandwidth that is persistently connected to the Internet **107**, while the Internet **107** connection for module **101** may be transient as module **101** changes between sleep and active states. Module controller **105***x* can provide the server-side logic for managing communications and con-

trolling module **101** using a module database **105***k*. Application interface **105***i* can provide functionality for communicating with external servers or nodes, such as, but not limited to, an application server **171** illustrated in FIG. **1***d*.

**[0086]** A module controller **101***x* and application interface **105***i* may be applications programmed in a language such as, but not limited to, C, C++, Java, or Python and could provide functionality to support M2M applications such as, but not limited to, remote monitoring of sensors and remote activation of actuators. Module controller **105***x* and application interface **105***i* could also be software routines, subroutines, linked libraries, or software modules, according to preferred embodiments. Many of the logical steps for operation of server **105**, module controller **105***x*, and/or application interface **105***i* can be performed in software and hardware by various combinations of physical interface **105***a*, system bus **105***d*, device driver **105***g*, and operating system **105***h*. A module controller **105***x* and application interface **105***i* can also access a set of cryptographic algorithms **141** (in FIG. **1***g* below) in order (i) to encrypt and decrypt data, and also (ii) process or generate a digital signature and verify received digital signatures. When server **105** is described herein as performing various actions such as, but not limited to, acquiring an IP address, monitoring a port, transmitting or sending a packet, receiving a message, or encrypting or signing a message, specifying herein that server **105** performs an action can refer to software, hardware, and/or firmware operating within server **105** performing the action. As contemplated herein, when a server **105** is described as performing an action such as, but not limited to, sending a response, receiving a message, verifying a digital signature, decrypting data, etc., in some embodiments a set of servers **105***n* can perform the actions for the server **105**. In this case, a server **105** could be a member of the set of servers **105***n*.

**[0087]** The server **105** may also include a user interface **105***j* such as a display (not shown) which could also comprise any type of display devices such as a liquid crystal display (LCD), a plasma display, and an organic light-emitting diode (OLED) display, or a cathode ray tube (CRT). A user interface **105***j* for the server **105** may optionally be provided remotely such as, but not limited to, (i) via a web browser or a secure terminal such as, but not limited to, secure shell (SSH) with (ii) another computer operated by an administrator (not shown). A user or administrator may enter commands and information into server **105** through a user interface **105***j*, such as, but not limited to, a keypad, keyboard, and a pointing device. In addition, the server **105** may store computer executable instructions such as, but not limited to, module controller **105***x* or application interface **105***i* on storage **105***m*. Storage **105***m* may comprise a disk drive, a solid-state drive, an optical drive, or a disk array. Module controller **101***x* (i) can manage communications with module **101** or a plurality of modules **101** and (ii) may be downloaded and installed on the server **105**. As noted previously and elsewhere herein, module program **101***i* and module controller **105***x* can preferably interoperate with each other in order to collect sensor data and control an actuator associated with a monitored unit **119**.

**[0088]** The application interface **105***i* and/or module controller **101***x* operating within server **105** illustrated in FIG. **1***c* can provide computer executable instructions to hardware such as CPU **105***b* through a system bus **105***d* in order to (i) receive a message from the module **101** and (ii) send a response, wherein the message can include sensor **101***f* data and the response can include an acknowledgement of the message and/or an instruction to the module **101**. The module controller **105***x* can enable the server **105** to send a response to a message from module **101** by recording data associated with module **101** in memory such as RAM **105***e*, where the data can include an instruction from module **101**, a destination IP:port number, a packet or packet header value, and the data can be processed using an encryption or ciphering algorithm or key, a digital signature algorithm or key, etc.

**[0089]** The application interface **105***i* can enable (a) the server **105** to send a datagram, packet, response to a module **101**, or an application message to an application server **171** (*b*) recording data associated (i) a with module **101** or (ii) other M2M service control information in memory such as RAM **105***e*, where the data can include information from module **101**, a destination IP:port number, a packet or packet header value, and the information could be processed using an encryption or ciphering algorithm or key, a digital signature algorithm or key, etc. The operating system **105***h* or the device driver **105***g* can write the data from RAM **105***e* to a physical interface **105***a* using a system bus **105***d* and an Ethernet connection in order to send the data via the Internet **107** illustrated in FIG. **1***a*. Alternatively, the software program **105***i* and/or module controller **105***x* can write the data directly to the physical interface **105***a* using the system bus **105***d*.

**[0090]** The server **105** can utilize the physical interface **105***a* to receive data from a module **101** and/or application **171***i* using a local area network such as Ethernet, although the physical interface **105***a* of server **105** could also utilize a wireless connection. The server **105** can listen or monitor for data from the Internet **107** using port number and/or a TCP/UDP socket. The received data from a module **101** can be a message formatted according to an Internet packet or datagram or series of datagrams inside Ethernet packets and include information from a module **101** such as, but not limited to, a source IP address and port number, an identity of the module, sensor data that may be encrypted, and/or a digital signature of the module. The received data from application **171***i* can comprise a series of datagrams formatted according to Internet Protocol and/or datagrams inside Ethernet packets. The received data or message from application **171***i* can include information regarding application **171***i* and/or server **105**, such as a source IP address and port number associated with application **171***i* and/or application server **171**, an identity of the server, actuator instructions or commands for a module **101** that may be encrypted, and a digital signature associated with the application **171***i*.

**[0091]** When server **105** receives messages or data, the operating system **105***h* or device driver **105***g* can record the received data from module **101** or application **171***i* via physical interface **105***a* into memory such as RAM **105***e*. The application interface **105***i* or operating system **105***h* may subsequently access the memory in order to process the data received. The application interface **105***i* and/or module controller **105***x*, or operating system **105***h* can include steps to process the data recorded in memory and received from the module **101** or application **171***i*, such as, but not limited to, parsing the received packet, decrypting data, verifying a digital signature with a key, or decoding sensor data included in a message from the module.

**[0092]** The server **105** and/or application interface **105***i* may communicate with application **171***i* by sending and receiving packets over a LAN or the Internet **107**, using a physical interface **105***a* and a wired connection such as Eth-

ernet or possibly a wireless connection as well. The server **105** can use the physical interface **105***a* such as an Ethernet connection to send and receive the data from the Internet **107**. For those skilled in the art, other steps are possible as well for an application interface **105***i* or operating system **105***h* within a server **105** to (i) send/receive a packet or message to/from a module **101** and (ii) send/receive a packet or message to/from an application **171***i* without departing from the scope of the present invention. Application interface **105***i* and module controller **105***x* may optionally be combined within a server **105**, or alternatively distributed across different physical computers and function in a coordinated manner using a network.

**[0093]** The device drivers **105***g*, operating systems **105***h*, and/or module controller **105***x* could also optionally be combined into an integrated system for providing the server **105** functionality. Although a single physical interface **105***a*, device-driver set **105***g*, operating system **105***h*, module controller **105***x*, application interface **105***i*, and user interface **105***j* are illustrated in FIG. **1***c* for server **105**, server **105** may contain multiple physical interfaces, device drivers, operating systems, software programs, module programs, and/or user interfaces. Server **105** may operate in a distributed environment, such that multiple computers operate in conjunction through a network to provide the functionality of server **105**. Also, server **105** may operate in a "virtualized" environment, where server **105** shares physical resources such as a physical CPU **105***b* with other processes operating on the same computer. And other arrangements could be used as well, without departing from the invention.

**[0094]** FIG. **1***d*

**[0095]** FIG. **1** *d* is a graphical illustration of hardware, firmware, and software components for an application server, in accordance with exemplary embodiments. Application server **171** can include application **171***i*. Application **171***i* can comprise a computer program or collection of computer programs, for managing a plurality of modules **101** using one or more servers **105**, including a set of servers **105***n* illustrated in FIG. **1***h* below. Application **171***i* can include a web portal **171***j*, service controller **171***x*, an application database **171***k*, and cryptographic algorithms **141**. During operation, such as when application **171***i* processes data from/to modules **101** through server **105**, application **171***i* may reside in RAM **171***e* within an application server **171**. Application **171***i* and the associated computer programs may be recorded in storage **171***m* so that they may be loaded by operating system **171***h* upon the startup of application server **171**. Web portal **171***j* can comprise a web server such as, but not limited to, Apache and can provide a user interface for a remote user accessing application **171***i* via an Internet **107**. The web portal **171***j* could include web pages for viewing reports from modules **101** and/or servers **105**, and also inputting settings for modules **101** by a user. The web pages could include PHP, active server pages, or Java components, in addition to other elements. Data input and stored by application **171***i* can be recorded in application database **171***k*. The data could be inserted or queried using structured query language (SQL) statements. Cryptographic algorithms **141** are depicted and described in connection with FIG. **1** *g* below.

**[0096]** Application **171***i* may be processed by an application server **171** using a CPU **171***b*. The illustrated components for the application server **171** in FIG. **1** *d* include a central processing unit (CPU) **171***b*, a random access memory (RAM) **171***e*, a system bus **171***d*, storage **171***m*, an operating system **171***h*, and an application **171***i*. These elements can provide functions equivalent to the central processing unit (CPU) **105***b*, RAM **105***e*, system bus **105***d*, storage **105***m*, and an operating system **105***h* described above in FIG. **1***c*, respectively. Application server **171** can comprise a general purpose computer such as, but not limited to, a rack mounted server within a data center or rack, or could also comprise a desktop computer or laptop. Application server **171** could also be a specialized computer, with hardware and software selected for supporting a plurality of servers **105** or modules **101** connecting and communicating simultaneously. Operating system **171***h* can comprise an operating system appropriate for a server such as, but not limited to, Linux, Solaris®, or Windows® Server. Application server **171** can preferably have a wired Ethernet connection with high bandwidth that is persistently connected to the Internet **107**.

**[0097]** An application **171***i* and/or service controller **171***x* may be an application programmed in a language such as, but not limited to, C, C++, Java, or Python and could provide functionality to support M2M applications such as, but not limited to, remote monitoring of sensors and remote activation of actuators. Application **171***i* can include a service controller **171***x*. Application **171***i* and/or service controller **171***x* could also be a software routine, subroutine, linked library, or software module, according to one preferred embodiment. Application **171***i* can include a service controller **171***x*, which can provide the functionality or CPU **171***b* instructions for the service controller **171***x* described in the present invention. Service controller **171***x* can include (i) logic for processing alarms from a module **101** (such as, but not limited to, sending out and email or text message to a user), (ii) logic for adjusting actuator **101***y* settings based upon data from sensor **101***f*, (iii) accepting user input (possibly via web portal **171***j*) and then making an associated change in an actuator **101***y* setting. Service controller **171***x* can also accept input from external applications (not shown) in order to make decisions regarding module **101**, sensor **101***f*, and/or actuator **101***y*.

**[0098]** Service controller **171***x* could be included within an enterprise resource planning (ERP) solution such as, but not limited to, SAP® or Oracle® ERP. An external application (not shown) can communicate with the application server **171**. As one example, a group of modules **101** could be installed within a manufacturing plant, and when a customer order was entered into the external application such as ERP, the service controller **171***x* could provide instructions for a group of modules **101** to server **105**, such as, but not limited to, changing actuators **101***y* to operate a production line. Other possibilities for service controller **171***x* exist as well without departing from the scope of the present invention. In general, service controller **171***x* can manage the overall function of a group of modules **101** through server **105**. Service controller **171***x* may operate at the "user layer" and/or "application layer" of the traditional OSI model.

**[0099]** Many of the logical steps for operation of application server **171** or application **171***i* can be performed in software by various combinations of physical interface **171***a*, device driver **171***g*, operating system **171***h*, and module controller **105***i*, where application **171***i* communicates with module controller **105***i* over a network. Application **171***i* and module controller **105***i* can communicate using an application message **701** (illustrated in FIG. **7** below). When application **171***i* is described herein as performing various actions such as, but not limited to, acquiring an IP address, monitoring a port, transmitting or sending a packet or message, or

encrypting or signing a message, receiving a packet or message, specifying herein that application **171***i* and/or application server **171** performs an action can refer to software, hardware, and/or firmware operating within application server **171** performing the action. Application server **171** or application **171***i* can send or transmit a message, packet, or data using the steps depicted and described in connection with FIG. **1***c* for a server **105** to send or transmit a message, packet, or data. Application server **171** or application **171***i* can receive a message, packet, or data using the steps depicted and described in connection with FIG. **1***c* for a server **105** to receive a message, packet, or data. Application server **171** can utilize hardware components similar to server **105**, such as storage **171***m* can be similar to storage **105***m*, CPU **171***b* can be similar to CPU **105***b*, and physical interface **171***a* can be similar to physical interface **105***a*. Application server **171** can use a system bus **171***d* to connect the hardware components shown within application server **171**, and system bus **171***d* can be similar to system bus **105***d* depicted and described in connection with FIG. **1***c* above.

**[0100]** Application server **171** may also comprise a collection of individual computers, where the individual computers could be either centrally located or geographically dispersed, but the individual computers may function in a coordinated manner over a network to operate as an application server **171**. In a similar manner, application **171***i* may be distributed across a plurality of computers, such as, but not limited to, in a cloud computing configuration. Application server **171** may be a "virtualized" server, with computing resources shared with other processes operating on a computer.

**[0101]** FIG. **1***e*

**[0102]** FIG. **1***e* is a graphical illustration of components within a module, in accordance with exemplary embodiments. FIG. **1***e* is illustrated to show a combination of components useful for leveraging the efficient and secure communication techniques described in the present invention. In addition to the components illustrated in FIG. **1***b* above, module **101** can include a battery **101***k*, a server public key **114**, a wireless module private key **112**, a connection to an actuator **101***y*, a USB interface **101***v*, a CPU wake controller **101***u*, a flash memory **101***w*, a symmetric key **127**, a pre-shared secret key **129***a*, a random number generator **128**, cryptographic algorithms **141**, a radio **101***z*, and other components illustrated in FIG. **1** *e*. Not all of the components illustrated in FIG. **1***e* are required for many exemplary embodiments, and some of the components illustrated in FIG. **1***e* may also be optionally omitted in exemplary embodiments.

**[0103]** The CPU **101***b* can comprise a general purpose processor appropriate for the low power consumption requirements of a module **101**, and may also function as a microcontroller. A CPU **101***b* and a CPU wake controller **101***u* are depicted and described in connection with FIG. **1***b* of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix, entitled "Systems and Methods for 'Machine-to-Machine' (M2M) Communications Between Modules, Servers, and an Application using Public Key Infrastructure (PKI)," which is hereby incorporated by reference in its entirety.

**[0104]** Sensor **101***f* could be a device to collect environmental data or data regarding a monitored unit **119**. Sensor **101***f* could collect data such as, but not limited to, temperature, humidity, pressure, visible light levels, radiation, shock and/or vibration, voltage, current, weight, pH levels, orientation/motion, or the presence of specific chemicals. Sensor **101***f* could also be a microphone. Sensor **101***f* could be a magnetic strip reader for credit cards and similar cards, or an antenna for either near-field RF communications, such as, but not limited to, reading an RF identity tag. An antenna for a sensor **101***f* could also collect longer-range RF signals, such as, but not limited to, reading long-range radio frequency identity tags. Sensor **101***f* could also collect biometric data such as, but not limited to, heart rate, glucose levels, body temperature, or other health measurements and in this case monitored unit **119** could be a person. The sensor **101***f* can provide data to the CPU **101***b* in the form of analog or digital data, which can be communicated via a system bus **101***d* or physical interface **101***a* and other electrical interfaces are possible as well. A sensor measurement can comprise the analog or digital data collected by CPU **101***b* from sensor **101***f*. A sensor measurement can include processing of the analog or digital data input CPU **101***b* by sensor **101***f*, such as, but not limited to, averaging over time, using mathematic formulas to convert the raw data from sensor **101***f* into a usable form. Module **101** may also collect sensor data or sensor values using a sensor **101***f* and CPU **101***b*, where the data or values are derived from electrical signals output by a sensor **101***f*. A sensor measurement can comprise the sensor data or sensor values. If module **101** comprises a "point of presence" payment terminal, then a sensor measurement could comprise data read from a payment card.

**[0105]** As contemplated herein, the terms "sensor measurement" and "sensor data" can be used interchangeably, and can also be considered functionally equivalent. Although a single sensor **101***f* is shown in FIG. **1***e*, a module **101** could include multiple sensors. Each of the multiple sensors **101***f* could include a sensor identity **151**, which could comprise a number or string to identify the sensor **101***f*. A sensor **101***f* could be external to module **101**, and also a plurality of sensors **101***f* may be used and they also can connect to module **101** when module **101** uses radio **101***z* as a base station for a WiFi network. An exemplary embodiment where sensor **101***f* connects to module **101** using a radio **101***z* is also depicted and described in connection with FIG. **1***e* of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix, which is hereby incorporated by reference in its entirety.

**[0106]** Actuator **101***y* could be a device to control a parameter or state for a monitored unit **119**, such as, but not limited to, changing a voltage or current, activating a switch or relay, turning on or off a microphone or speaker, activating or deactivating a light, and other examples are well known in the art. Actuator **101***y* could also be a speaker. Actuator **101***y* could be controlled by module **101** via a digital or analog output from CPU **101***b*, which could also be transmitted or sent via system bus **101***d* or a physical interface **101***a*. Although actuator **101***y* is illustrated as external to wireless module **101** in FIG. **1***e*, actuator **101***y* could also be internal to module **101**, and module **101** could include multiple actuators **101***y*. The use of multiple actuators **101***y* each with an actuator identity **152** is also depicted and described in connection with FIG. **1***e* of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix, which is hereby incorporated by reference in its entirety. Sensors and actuators are well known to those of ordinary skill in the art, and thus are not described in additional detail herein.

**[0107]** Module **101** can include a Universal Serial Bus (USB) interface. In accordance with an exemplary embodiment, module **101** can comprise a wireless module and

include a radio **101***z*. Note that the use of a radio **101***z* is not required for module **101**, which could also obtain a connection to the Internet **107** via a wired line such as Ethernet. Although not illustrated, radio **101***z* could include antennas for reception and transmission of RF signals, and even multiple antennas could be used. Although a single radio **101***z* is illustrated in module **101**, module **101** could also contain multiple radios **101***z*. Radio **101***z* can support wireless LAN standards such as, but not limited to, WiFi, Bluetooth, and Zigbee, or similar wireless LAN standards. Radio **101***z* illustrated in FIG. **1***e* can comprise a radio **101***z* depicted and described in connection with FIG. **1***d* of U.S. patent application Ser. No. 14/039,401, filed Sep. 27, 2013 in the name of John Nix, the contents of which are herein incorporated in their entirety.

**[0108]** Note that module **101** may also operate as a base station in a wireless LAN, such as, but not limited to, an 802.11 base station. When module **101** operates a wireless LAN, radio **101***z* can function as either a client/node and/or a base station **103** to support communication from other wireless nodes in physical proximity, such as, but not limited to, other nodes within an exemplary 50 meters. The other wireless nodes could comprise a sensor **101***f* and/or actuator **101***y*, and in this case a sensor could be referred to as a "networked sensor" and an actuator could be referred to as a "networked actuator". Radio **101***z* functioning as a base station is depicted and described as a base station **103** is depicted and described in connection with FIG. **1***d* of U.S. patent application Ser. No. 14/039,401, filed Sep. 27, 2013 in the name of John Nix, the contents of which are herein incorporated in their entirety.

**[0109]** In accordance with exemplary embodiments, module **101** can store module private key **112**, server public key **114**, and module identity **110**, and a symmetric key **127** in memory/RAM **101***e* during operation, such as when CPU **101***b* is active and the module **101** is connected to a network such as a wireless network **102** during data transmissions. Module private key **112** preferably is recorded in nonvolatile memory such as, but not limited to, flash memory **101***w*, so that module **101** has access to its private key **112** after the private key has been derived or loaded, including times when a battery **101***k* has been fully drained or removed from module **101** (if module **101** does not utilize a persistent power source such as land-line power).

**[0110]** Symmetric key **127** can be a secure, shared private key for use with symmetric encryption or symmetric ciphering algorithms **141***b* (in FIG. **1***g* below). Symmetric key **127** can be derived by using module public key **111** and/or server public key **114**, possibly through the use of a key derivation function **141***f* (described in FIG. **1***g* below). Symmetric key **127** can be used for both encryption and decryption with symmetric cryptographic algorithms **141***b* described in FIG. **1***g* below, where a shared secret key can be used to encrypt/cipher and/or decrypt/decipher. Symmetric key **127** may also include an expiration time **133**, such that symmetric key **127** may only be used by module **101** and/or server **105** during a limited period of time, such symmetric key **127** remaining only valid for a day, or a week, or during a session (where the session comprises multiple messages and/or responses between a module **101** and a server **105**), etc. Module **101** can also derive symmetric key **127** according the Elliptic Curve Integrated Encryption Scheme (ECIES) and/or ECDH **159**, discussed in FIG. **1***g* below, using module public key **111**, server public key **114**, and a random number **128***a* from random number generator **128**. ECIES could be included in cryptographic algorithms **141**. A summary of ECIES shared key derivation is described the Wikipedia article "Integrated Encryption Scheme" from Sep. 18, 2013 (herein incorporated by reference). Other possibilities for shared key derivation function using public keys are possible as well, including a Diffie-Hellman key exchange. Using a derived symmetric key from the exemplary key derivation function ECIES, module **101** and/or server **105** could derive a second symmetric key **127** after the expiration time **133** of the first symmetric key **127** had transpired. As contemplated herein, a symmetric key **127** can also comprise a session key, or the use of a "session key" with a symmetric ciphering algorithm **141***b* can comprise a symmetric key **127**.

**[0111]** Note that a key derivation function **141***f* using public keys is not required to generate a shared symmetric key **127**, and alternatively a shared symmetric key **127** could be generated by any of module **101**, server **105**, module provider **109**, M2M service provider **108**, or application server **171**. If module **101** generates shared symmetric key **127** for symmetric ciphering **141***b* within a cryptographic algorithms **141**, then module **101** can send shared symmetric key **127** to server **105** using an asymmetric ciphering depicted and described in connection with FIG. **4** below. In accordance with a preferred exemplary embodiment, module **101** preferably uses a random number generator **128** to generate a random number **128***a* (illustrated in FIG. **1***g*) for input into cryptographic algorithms **141**, and the seed **129** in random number generator **128** could utilize data from a sensor **101***f* in order to generate a random number **128***a* with high entropy in the creation of symmetric key **127**. Random number generator **128** and random number **128***a* are also depicted and described in connection with FIG. **1***d* of U.S. patent application Ser. No. 14/039,401, filed Sep. 27, 2013 in the name of John Nix, the contents of which are herein incorporated in their entirety.

**[0112]** Module identity **110** is preferably a unique identifier of module **101**, and could comprise a number or string such as, but not limited to, a serial number, an international mobile subscriber identity number (IMSI), international mobile equipment identity (IMEI), or an Ethernet media access control (MAC) address. According to an exemplary embodiment, module identity **110** can also comprise a serial number or string that is written into hardware of module **101** upon manufacturing or distribution of module **101**. In this case, module identity **110** could be recorded in a read only memory **101***c*, where read only memory **101***c* could not be easily erased or otherwise tampered with. Read only memory **101***c* could also comprise a protected memory. Or, module **101** could read module identity **110**, which could be written into hardware by a manufacturer, distributor, or module provider **109**, by using a device driver **101***g* that reads a hardware address containing the module identity **110** using the system bus **101***d*. Module **101** can read the module identity **110** by accessing a read-only address using the bus **101***d*. In either case, in one embodiment module identity **110** may preferably be permanently or persistently associated with the physical hardware of module **101**, which can be helpful for the security procedures contemplated herein. Module identity **110** can function as a basic identifier for services from M2M service provider **108**, server **105**, and/or application **171***i* in order to properly identify module **101** among a plurality of modules. Module private key **112** and module public key **111** could be unique to module **101** and uniquely associated with module identity **110**, according to a preferred embodiment.

**[0113]** As contemplated herein, a module identity **110** can also have more than one use. A first module identity **110** could comprise a serial number for the physical hardware of module **101**, as described in the paragraph above. A second module identity **110** could also comprise a session identifier, for data sessions between module **101** and server **105**, where the session identifier can be uniquely associated by a server **105** to module **101**. In the case where module identity **110** has more than one use, format, or representation, the module identity **110** associated with or written into hardware of module **101** (and potentially read from a read-only address in module **101**) would preferably comprise the module identity **110** used in a certificate **122**. Since a module **101** may utilize multiple module public keys **111** and module private keys **112** over its lifetime, a certificate **122** for module **101** can preferably include both (i) the module identity **110** (such as, but not limited to, a serial number for the physical hardware of module **101**) and (ii) a module public key identity **111***a* in order to specify the particular module public key **111** associated with certificate **122**. The use of a module public key identity **111***a* in a certificate **122** is also depicted and described in connection with FIG. **1***h* of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix. Since a module **101** may also have multiple public keys **111** for different purposes (such as, but not limited to, one for creating digital signatures, another for asymmetric ciphering, another for use with a second wireless network **102**, etc.), then module **101** may also potentially have multiple valid certificates **122** concurrently each with different module public key identities **111***a*.

**[0114]** Further, as contemplated herein, a module identity **110** could also comprise more than one physical string or number, such as, but not limited to, a first string when module **101** connects with a first M2M service provider **108** or first wireless network **102**, and module identity **110** could comprise a second string when module **101** connects with a second M2M service provider **108** or second wireless network **102**. The first M2M service provider **108** or first wireless network **102** may have a first requirement or specification for the format, length, structure, etc. of module identity **110**, and the second M2M service provider **108** or second wireless network **102** may have a second requirement or specification for the format, length, structure, etc. of module identity **110**.

**[0115]** Server public key **114** in module **101** could be obtained from downloading the key over the Internet **107**, or optionally also written into nonvolatile memory of module **101** upon manufacture or distribution. Server public key **114** could be obtained using a domain name or Internet address that is recorded in nonvolatile memory upon the configuration of module **101**, such as, but not limited to, during installation or distribution, and module **101** could fetch the server public key **114** upon connecting to a wireless network **102** or other connection to the Internet **107**.

**[0116]** Module **101** may also contain cryptographic algorithms **141**, which may comprise a suite of algorithms or subroutines that can be utilized for (i) deriving a pair of keys comprising a public key and a private key, (ii) encrypting data using public keys, (iii) decrypting data using private keys, (iv) processing secure hash signatures using private keys, and (v) verifying secure hash signatures using public keys, and related software, firmware, or subroutines for implementing a cryptographic system, including symmetric ciphering algorithms. Cryptographic algorithms **141** (also described below in FIG. **1***g*) could utilize publicly available software libraries within tools such as, but not limited to, OpenSSL maintained by The OpenSSL Project (http://www.openssl.org/), libgcrypt maintained by The Free Software Foundation (http://www.gnu.org/software/libgcrypt/), and similar libraries such as, but not limited to, libmcrypt and Crypto++. Note that cryptographic algorithms **141** could also use proprietary cryptographic libraries as well. In addition to implementing asymmetric encryption/ciphering, such as, but not limited to, used with RSA and ECC cryptography, cryptographic algorithms **141** can provide symmetric ciphering where a shared private key is utilized to both encrypt and decrypt, such as, but not limited to, with the Advanced Encryption Standard (AES) cipher suite.

**[0117]** As illustrated in FIG. **1***e*, module **101** may also contain a random number generator **128**. Random number generator **128** may contain a seed **129**. The creation of random numbers with a high degree of entropy may be important the use of cryptographic algorithms **141**. A plurality of the data as a source for a random number seed **129** could be appended together into a "module random seed file" **139** (illustrated in FIG. **1***g*) with a combined series or list of states (i.e. a plurality of sensor **101***f* measurements, radio **101***z* measurements, clock times, memory **101***e* or memory **101***w* states, operating system **101***h* states, actuator **101***y* states, and/or hardware **101***a* or **101***d* states). Note that values or data for each of the elements listed in the previous sentence could be utilized in a "module random seed file" **139** instead of or in addition to a state. The "module random seed file" is also depicted and described in connection with FIG. **1***e* of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix, which is hereby incorporated by reference in its entirety

**[0118]** Note that the term "public key" as contemplated herein includes a key that may be shared with other elements, where the other elements may not be under the direct control of the same entity that holds the corresponding private key. However, the term "public key" as used herein does not require that the public key is made available to the general public or is publicly disclosed. An additional layer of security may be maintained in the present invention by preferably only sharing public keys on a confidential basis with other entities. For example, module public key **111** may be created by module **101** when generating module private key **112**, and module **101** may share module public key **111** with M2M service provider **108** in order to record module public key **111** in server **105**, but module **101** could choose to not share module public key **111** with other entities, such as wireless network **102** or provide a certificate **122** with module public key **111** publicly available on the Internet **107**. The benefits of confidentially sharing module public key **111** with server **105** are also further described below.

**[0119]** Although a single public key and private key for (i) module **101** and (ii) server **105** are illustrated in FIG. **1***e* and also FIG. **1***f* below, respectively, both module **101** and server **105** may each utilize several different pairs of public keys and private keys. As one example, module **101** may record a first private key **112** used for creating a digital signature and a second private key **112** for decryption using asymmetric ciphering algorithms **141***a*. In this example, a server **105** could utilize a first module public key **111** to verify the digital signature, and a second module public key **111** could be utilized to encrypt messages sent to module **101**. Similarly, either module **101** or server **105** may use private key **112** or **105***c*, respectively, to derive secondary shared keys such as,

but not limited to, a derived shared key **129***b* below. Thus, one key pair could be used with digital signatures, a second key pair used for asymmetric ciphering, and a third key pair to derive shared secret keys. Each of the three illustrated pairs of keys could comprise a set of keys, and each of the illustrated pairs of keys could also use a different set of cryptographic parameters **126** (illustrated in FIG. **1** *g* below), although the cryptographic parameters **126** for the various pairs of keys could also be the same.

**[0120]** In addition, module **101** could utilize a first set of keys to communicate with a first server **105** and a second set of keys to communicate with a second server **105**. The first set of keys could use or be associated with a first set of cryptographic parameters **126** and the second set of keys could use or be associated with a second set of cryptographic parameters **126**. According to exemplary embodiments, module **101** may also include a pre-shared secret key **129***a*. Pre-shared secret key **129***a* can comprise a secret key that is shared between module **101** and server **105** before module **101** begins (i) communicating with server **105** and/or a certificate authority **118**, (ii) or utilizing PKI-based encryption and authentication to communicate with M2M service provider **108**. As illustrated in FIG. **1***f* below, server **105** could also record the pre-shared secret key **129***a*, and a pre-shared secret key **129***a* can be associated with each module **101** using a module identity **110**. A pre-shared secret key **129***a* is also depicted and described in connection with U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix, which is hereby incorporated by reference in its entirety. In an exemplary embodiment, once the pre-shared secret key **129***a* has been utilized to authenticate or verify a module public key **111** with a server **105** (such as, but not limited to, using subsequent steps **517** in FIG. **5***b* below), then that particular pre-shared secret key **129***a* may be "discarded" and not used again for security purposes contemplated herein.

**[0121]** Note that the use of a pre-shared secret key **129***a* and pre-shared secret key code **134** is also optional, such that a module program **101***i* could cipher of obfuscate the initial submission of a derived module public key **111** and module identity to a server **105**, so that server **105** could be reasonably assured only a valid module **101** submitted the module public key **111**. According to a preferred exemplary embodiment, module **101** can derive its own module private key **112** and module public key **111**, and utilize pre-shared secret key **129***a* in order to securely and/or authoritatively communicate the derived module public key **111** with server **105** and/or a certificate authority **118**. The use of pre-shared secret key **129***a* can be particularly useful if module **101** has already been deployed with a monitored unit **119** and connects to server **105** though the Internet **107** for the very first time. Server **105** could preferably utilize pre-shared secret key **129***a* in order to confirm that a received module public key **111** and module identity **110** from module **101** authoritatively belong to module **101**, as opposed to being an unauthorized or even fraudulent submission of module public key **111** and module identity **110**.

**[0122]** Server **105** could utilize a pre-shared secret key **129***a* and the steps depicted and described in connection with FIG. **4** below in order to securely receive module public key **111** and module identity **110** from module **101**, including the first time module **101** sends module public key **111** to server **105**. As one example, pre-shared secret key **129***a* could be utilized as a symmetric ciphering **141***b* key, described in FIG. **1** *g* below. After the first submission of module public key **111** to server **105**, any subsequent submissions of new module public keys **111** derived by module **101** could either (i) continue to use the pre-shared secret key **129***a*, or (ii) use a symmetric key **127** derived after the first module public key **111** has been received. Securing the submission of module public key **111** with server **105**, including both the first submission and subsequent submissions, is also depicted and described in connection with FIG. **5***b* below.

**[0123]** FIG. **1***f*

**[0124]** FIG. **1***f* is a graphical illustration of components within a server, in accordance with exemplary embodiments. Server **105** can include a module database **105***k*, a sub-server **105***w*, and a message preprocessor **105***y*. In an exemplary embodiment, the elements illustrated within a server **105** in FIG. **1***f* may be stored in volatile memory such as RAM **105***e*, and/or storage **105***m*, and may also be accessible to a processor CPU **105***b*. In another exemplary embodiment, the module database **105***k*, sub-server **105***w*, and message processor **105***y* can comprise separate computers. Module database **105***k*, sub-server **105***w*, and message preprocessor **105***y* could represent either different processes or threads operating on a server **105**, or physically separate computers operating in conjunction over a network to perform the functions of a server **105**. Since server **105** can preferably support communications with a plurality of modules **101**, server **105** can utilize module database **105***k* to store and query data regarding a plurality of modules **101**, monitored units **119**, and the overall M2M service. The server **105** can store a plurality of module public keys **111** associated with a plurality of devices in the module database **105***k*. The server **105** can use the module identity **110** of device **101**, received in a message such as, but not limited to, a UDP packet, to query the module database **105***k* and select the public key **111** or symmetric key **127** associated with the module **101**.

**[0125]** Although not illustrated in FIG. **1***f*, module database **105***k* can also record a pre-shared secret key code **134**, a set of cryptographic parameters **126**, and a module identity **110** for each module **101**, along with the pre-shared secret key **129***a* shown in FIG. **1***f*. In addition, although not illustrated in FIG. **1***f*, module database **105***k* could store a symmetric key **127** for each module **101**, if cryptographic algorithms **141** utilize a symmetric cipher **141***b* such as, but not limited to, AES for communication with module **101**. Examples of module database **105***k* could include MySQL, Oracle®, SQLite, hash tables, distributed hash tables, text files, etc. Module database **105***k* could reside within RAM **105***e* or storage **105***m*. Server **105** may also record a symmetric key **127**, where the symmetric key **127** can be associated with an expiration time **133**. Symmetric key **127** can also be recorded in a module database **105***k* or a sub-server **105***w*.

**[0126]** Message preprocessor **105***y* can process incoming packets and route them to an appropriate sub-server **105***w* using information contained in an incoming message, such as, but not limited to, a module identity **110**, a server identity **206** illustrated in FIG. **2** below, and/or a destination IP address. Message preprocessor **105***y* can include rules for processing and routing, such a dropping malformed incoming messages or incoming messages without correct cryptographic data. Message preprocessor **105***y* could also optionally be combined with a server firewall **124** in order to provide firewall functionality and security at the network layer. Message preprocessor **105***y* may preferably remain "silent" to incoming packets without proper cryptographic data con-

tained in an incoming message, such as, but not limited to, one example of a properly formatted message **208** illustrated in FIG. **6***a* below.

**[0127]** Sub-server **105***w* can include a server private key **105***c* and cryptographic algorithms **141**. A plurality of sub-servers **105***w* can be utilized by a server **105** in order to support communication with a plurality of modules **101**. The server private key **105***c* and module public key **111** can be utilized by server **105** to secure communication with module **101**, including the steps depicted and described in connection with FIG. **4** and FIG. **5***a* below. Cryptographic algorithms **141** may comprise a suite of algorithms or subroutines and are depicted and described in connection with FIG. **1***g*.

**[0128]** A first sub-server **105***w* can process messages and responses with a first module **101** using a first set of security keys and algorithms, such as, but not limited to, using RSA-based security, and a second sub-server **105***w* can process messages and responses with a second module **101** using a second set of security keys and algorithms, such as, but not limited to, using ECC-based security. Consequently, message pre-processor **105***y* could route incoming messages to the appropriate sub-server **105***w* depending on the encryption algorithm used in the incoming message (which could be determined by message pre-processor **105***y* by querying the module database **105***k* using a module identity **110** in the incoming message **208**, where module identity **110** can be used to select a sub-server **105***w*). Sub-servers **105***w* may utilize separate server private keys **105***c*, or the sub-servers **105***w* can share a common private key **105***c*. Sub-servers **105***w* may utilize separate cryptographic algorithms **141**, or the sub-servers **105***x* can share common cryptographic algorithms **141**. Although separate sub-servers **105***w* are illustrated in FIG. **1***f*, the sub-servers may optionally be combined with a server **105**, or omitted, with the corresponding server private key **105***c* and cryptographic algorithms **141** stored directly in a server **105**.

**[0129]** Server **105** may also comprise a collection of individual computers, where the individual computers could be either centrally located or geographically dispersed, but the individual computers may function in a coordinated manner over a network to operate as a server **105**. Server **105** may be a "virtualized" server, with computing resources shared with other processes operating on a computer.

**[0130]** FIG. **1***g*

**[0131]** FIG. **1** *g* is a graphical illustration of components in a set of cryptographic algorithms, in accordance with exemplary embodiments. As contemplated herein, communications between (i) a module **101** and a server **105**, and (ii) between application **171***i* and server **105** can be secured by using cryptographic algorithms **141**. The cryptographic algorithms **141** used by module **101**, server **105**, application server **171**, and/or application **171***i* can comprise a set of steps, procedures, or software routines for accomplishing steps to cipher, decipher, sign, and verify messages, including the generation of public keys, private keys, and derived shared keys. Cryptographic algorithms **141** can be implemented in software operating on (i) module **101** in the form of a module program **101***i*, (ii) server **105** in the form of a module controller **105***x*, or (iii) application server **171** in the form of an application **171***i*. Example software for a cryptographic algorithms **141** includes the libraries within the openssl, libmcrypt, and/or and Crypto++ discussed in FIG. **1***e*. Other possibilities for the location of cryptographic algorithms within a module **101**, server **105**, or application **171***i* exist as well, including possibly module operating system **101***h*, server operating system **105***h*, and application server operating system **171***h*, respectively.

**[0132]** In addition, cryptographic algorithms **141** may be implemented in hardware or firmware on any of module **101**, server **105**, or application **171***i*. Note that module **101**, server **105** and application **171***i* could each utilize a different set of cryptographic algorithms **141**, although the sets of algorithms should preferably be fully interoperable (i.e. ciphering with a first symmetric ciphering algorithm **141***b* and a symmetric key **127** on module **101** could be deciphered by a second symmetric ciphering algorithm **141***b* on server **105** using the symmetric key **127**, etc.). As illustrated in FIG. **1***g*, cryptographic algorithms **141** may comprise an asymmetric ciphering algorithm **141***a*, a symmetric ciphering algorithm **141***b*, a secure hash algorithm **141***c*, a digital signature algorithm **141***d*, a key pair generation algorithm **141***e*, a key derivation function **141***f*, and a random number generator **128**.

**[0133]** Asymmetric ciphering algorithms **141***a* can comprise algorithms utilizing public key infrastructure (PKI) techniques for both (i) encrypting with a public key and (ii) decrypting with a private key. Example algorithms within asymmetric algorithms **141***a* include the RSA algorithms **153** and the Elliptic Curve Cryptography (ECC) algorithms **154**, and other asymmetric algorithms could be utilized as well. For example, either the ECC algorithms **154** or RSA algorithms **153** can be used for encryption and decryption, including (i) encryption step **503** discussed below, as well as (ii) decryption step **413** discussed below. A set of cryptographic parameters **126** can include input into asymmetric ciphering algorithms **141***a*, such as, but not limited to, specifying key lengths, elliptic curves to utilize (if ECC), modulus (if RSA) or other parameters or settings required. As contemplated herein and described in additional detail below, the algorithms illustrated in FIG. **1** *g* can perform both ciphering and deciphering, using the appropriate keys.

**[0134]** The use and application of RSA algorithms and cryptography are described within IETF RFC 3447 titled "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", herein incorporated by reference, among other published standards for the use of RSA algorithms **153**. The use of an RSA algorithm **153** for encryption and decryption, including with cryptographic algorithm and other description of encryption or decryption algorithms, can also be processed according to the description of the RSA algorithm according to the Wikipedia entry for "RSA (algorithm)" as of Sep. 9, 2013, which is incorporated by reference herein.

**[0135]** The use and application of ECC algorithms **154** for asymmetric ciphering algorithms **141***a* within cryptographic algorithms **141** are described within IETF RFC 6090 titled "Fundamental Elliptic Curve Cryptography Algorithms" (herein incorporated by reference), among other published standards using ECC. ECC algorithms **154** can also utilize elliptic curve cryptography algorithms to the Wikipedia entry for "Elliptic curve cryptography" as of Sep. 9, 2013, which is incorporated by reference herein. ECC algorithms **154** may utilized according to exemplary preferred embodiments in order to maintain high security with smaller key lengths, compared to RSA, thereby helping to comparably reduce the message lengths, radio frequency spectrum utilization, and processing power required by module **101**. Thus, the use of ECC algorithms **154** within various steps requiring ciphering or digital signatures may help conserve battery life of module

**101** while maintaining the objective of securing system **100**. Note that as contemplated herein, other algorithms besides with ECC algorithms **154** and RSA algorithms **153** may be also be used in asymmetric algorithms **141***a*.

**[0136]** Cryptographic algorithms **141** may also include a set of symmetric ciphering algorithms **141***b*. Symmetric ciphering algorithms **141***b* can utilize a symmetric key **127** by one node such as a module **101** to encrypt or cipher data, and the encrypted data can be decrypted or deciphered by server **105** also using the symmetric key **127**. Examples of symmetric ciphers include Advanced Encryption Standard 155 (AES), as specified in Federal Information Processing Standards (FIPS) Publication 197, and Triple Data Encryption Standard (Triple DES), as described in NIST Special Publication 800-67 Revision 1, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher (Revised January 2012)". Cryptographic parameters **126** input into symmetric ciphering algorithms **141***b* can include symmetric key **127** length, such as, but not limited to, the selection of 128, 192, or 256 bits with AES **155** symmetric ciphering, and cryptographic parameters **126** could also select a symmetric ciphering algorithm in a collections of symmetric ciphering algorithms **141***b*. Other examples of symmetric ciphering algorithms **141***b* may be utilized as well within cryptographic algorithms **141**. Also note that as contemplated herein, the term "symmetric ciphering" contemplates the use of a symmetric ciphering algorithm **141***b* in order to encrypt or cipher data with a symmetric ciphering algorithm **141***b*, and "asymmetric ciphering" contemplated the use of an asymmetric ciphering algorithm **141***a* to encrypt or cipher data with a public key, such as module public key **111** or server public key **114**.

**[0137]** Cryptographic algorithms **141** may also include a set of secure hash algorithms **141***c* in order to compute and output a secure hash value or number based on a string or file input into the secure hash algorithms **141***c*. Example secure hash algorithms include SHA256 156 (also known as SHA-2) and SHA-3 157. SHA256 156 is specified in the National Institute of Standards and Technology (NIST) Federal Information Processing Standards Publication (FIPS PUB) 180-2 titled "Secure Hash Standard". SHA-3 157 is scheduled to be published in FIPS PUB 180-5. Cryptographic parameters **126** input into secure hash algorithms **141***c* can include the selection of the length of the secure hash, such as, but not limited to, using 224, 256, or 512 bits with either SHA-2 or SHA-3, and other possibilities exist as well.

**[0138]** Cryptographic algorithms **141** may also include a set of digital signature algorithms **141***d*, in order to sign and verify messages between (i) module **101** and server **105** or (ii) server **105** and application **171***i*. Digital signature algorithms **141***d* can also verify signatures such as, but not limited to, comparing that (i) a first secure hash value in the form of a digital signature in a certificate (not shown) using a certificate authority public key matches (ii) a second secure hash value in the certificate (not shown). Digital signature algorithms **141***d* can utilize algorithms in National Institute of Standards (NIST) "FIPS 186-4: Digital Signature Standard", or IETF RFC 6979 titled "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)". The use of ECDSA algorithm **158** within a set of digital signature algorithms **141***d* may be preferred if keys such as, but not limited to, module public key **111** and server public key **114** are based on elliptic curve cryptography. Other PKI standards or proprietary techniques for securely verifying digital signatures may be utilized as well in digital signature algorithms **141***d*. Cryptographic parameters **126** input into digital signature algorithms **141***d* can include the selection of a secure hash algorithms **141***c* to utilize with digital signature algorithms **141***d*, or the algorithm to utilize, such as, but not limited to, ECDSA shown or an RSA-based alternative for digital signatures is possible as well. Cryptographic parameters **126** input into digital signature algorithms **141***d* can also include a padding scheme for use with a digital signature algorithms **141***d*. Digital signature algorithms **141***d* could also include an RSA digital signature algorithm for use with RSA-based public and private keys.

**[0139]** Cryptographic algorithms **141** may also include key pair generation algorithms **141***e*, a key derivation function **141***f*, and a random number generator **128**. Key pair generation algorithms **141***e* can be utilized by module **101**, server **105**, or application **171***i* to securely generate private and public keys. The key pair generation algorithms **141***e* can also use input from a cryptographic parameters **126**, such as, but not limited to, the desired key lengths, or a value for an ECC curve if the public key will support ECC algorithms **154**. According to an exemplary preferred embodiment, module **101** can derive a pair of module public key **111** and module private key **112** using key pair generation algorithms **141***e*. Software tools such as, but not limited to, openssl and libcrypt include libraries for the generation key pairs, and these and similar libraries can be used in a key pair generation algorithm **141***e*.

**[0140]** Key derivation function **141***f* can be used by module **101**, server **105**, and/or application **171***i* in order to determine a common derived shared secret key **129**, using at least two respective public keys as input, and may also include the input of a private key. A key exchange to share a common symmetric key **127** can be performed using a key derivation function **141***f* and cryptographic parameters **126**. An exemplary algorithm within a key derivation function **141***f* can be the Diffie-Hellman key exchange, which is used by tools such as, but not limited to, secure socket layer (SSL) with RSA algorithms **153**. When using ECC algorithms **154**, module **101** and server **105** can utilize Elliptic Curve Diffie-Hellman (ECDH) algorithms **159**, and a summary of ECDH is included in the Wikipedia article titled "Elliptic Curve Diffie-Hellman" (http://en.wikipedia.org/wiki/Elliptic_curve_Diffie % E2%80%93Hellman" from Sep. 24, 2013, which is herein incorporated by reference. Other algorithms to derive a shared secret key **129***b* using public keys and a private key may also be utilized in a key derivation function **141***f*, such as, but not limited to, the American National Standards Institute (ANSI) standard X-9.63 160. Cryptographic parameters **126** used with key derivation function **141***f* with elliptic curve cryptography can include a common base point G for two node using the key derivation function **141***f* and public keys. The base point G in a cryptographic parameters **126** can be transmitted or sent from a module **101** to a server **105** in a message **208**, and the base point G can be sent from a server **105** to a module **101** in a response **209**, and other possibilities exist as well. Cryptographic parameters **126** can also include other or additional information for using a key derivation function **141***f* in order to derive a commonly shared symmetric key **127**.

**[0141]** Cryptographic parameters **126** input into key pair generation algorithms **141***e* can include the type of asymmetric ciphering algorithms **141***a* used with the keys, the key length in bits, an elliptic curve utilized for ECC, a time-to-live

for a public key that is derived, and similar settings. Additional cryptographic parameters **126** for a public key can include 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 public key can have several formats and a set of cryptographic parameters **126** can be useful to specify the format. Although a set of cryptographic parameters **126** is illustrated in FIG. **1***g* as internal to cryptographic algorithms **141**, parameters **126** could be recorded in other locations in a module **101** or a system **100**. As one example, a set of cryptographic parameters **126** could be recorded in a server **105** and downloaded by module **101** using the Internet **107**. The various algorithms within cryptographic algorithms **141** may utilize a random number generator **128**, which is also depicted and described in connection with Figure le above. As contemplated herein, the term "cryptographic parameters" **126** may be considered equivalent to a "set of cryptographic parameters" **126**, and also use of the terms "parameters" **126** and "set of parameters" **126** can both refer to the cryptographic parameters **126** illustrated in FIG. **1***g*.

**[0142]** According to a preferred exemplary embodiment, cryptographic parameters **126** can include values to define an elliptic curve and/or use ECC algorithms **154**. A set of ECC parameters **137** could comprise values or numbers for an elliptic curve defining equation. ECC parameters **137** are also depicted and described in FIG. **1***g* of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix, which is hereby incorporated by reference in its entirety. Cryptographic parameters **126** could also include an ECC standard curve **138**, which could comprise a name and/or values for a standardized curve, such as, but not limited to, the list of named curves included in section 5.1.1 of IETF RFC 4492 titled "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)."

**[0143]** As contemplated herein, a set of cryptographic algorithms **141** may operate using either strings or numbers, and cryptographic parameters **126** could include either strings or numbers as well. As contemplated herein (i) a collection, sequence, and/or series of numbers could comprise a string, (ii) a string can include a mixture of numbers and characters, or (iii) a string can comprise a collection, sequence, and/or series of characters. The processing of cryptographic algorithms within a module **101** can take place within a CPU **101***b*, or module **101** could also process cryptographic algorithms in a cryptographic processing unit (not shown) connected to the system bus **101***d*. According to an exemplary embodiment, a module **101** or a server **105** could include a cryptographic processing unit (not shown) separate from the CPU **101***b* or CPU **105***b* in order to increase efficiency for supporting the use of cryptography through a system **100**. Alternatively, in exemplary embodiments cryptographic algorithms **141** can be implemented entirely in software within a module **101** and/or server **105**, and also utilized by a module controller **101***x* and application interface **101***i*.

**[0144]** FIG. **1***h*

**[0145]** FIG. **1***h* is a graphical illustration of an exemplary system that includes a user, an application, a set of servers, and a set of modules, in accordance with exemplary embodiments. System **199** illustrated in FIG. **1***h* can include a user **183**, an application **171***i*, a set of servers **105**, and a set of modules **101**, which can communicate as illustrated using the Internet **107**. Each of a server A **105** and server B **105** and additional servers can communicate with a plurality of modules. An application **171***i* can communicate with a plurality of servers **105**. Although the servers **105** and application **171***i* in system **100** in FIG. **1***i* are illustrated as being separate, application **171***i* and server **105** may optionally be combined into a single node, such that the application **171***i* and server **105** operate as separate processes or programs on the same computer, or on a computer operating in a distributed environment such as, but not limited to, a cloud configuration. In addition, even though a single application **171***i* and a single user **183** are illustrated in FIG. **1***i*, a system **199** could include multiple applications **171***i* and multiple users **183**. In exemplary embodiments, application server **171** and/or application **171***i* can communicate with multiple servers **105** using the multiple secure connection data transfer **802** links illustrated in FIG. **1***h*. A secure connection data transfer **802** is depicted and described in connection with FIG. **8** below. In addition, in exemplary embodiments, a first server A **105** can communicate with a second server B **105** also by using a secure connection data transfer **802**.

**[0146]** As illustrated in FIG. **1***h*, a first server A **105** and a second server B **105** could also share a module database **105***k*, such that information recorded in a module database **105***k* by the first server A **105** could be accessible to or queried by a second server B **105**. The connection between the servers and the module database **105***k* could also be through a secure connection data transfer **802** which is depicted and described in connection with FIG. **8**. In this manner and as contemplated herein, a module database **105***k* can comprise a shared module database **105***k*. Other configurations are possible as well without departing from the scope of the present invention for using a shared module database **105***k*. In another embodiment a shared module database **105***k* could be combined with application server **171** or shared module database **105***k* could comprise a distributed hash table. In exemplary embodiments, a system **199** could also include a plurality of shared module databases **105***k*, wherein the shared module databases **105***k* could be either (i) be periodically synchronized, or (ii) record separate information for a system **199**. A shared module database **105***k* could also comprise a plurality of separate module databases **105***k*, such that different information may be recorded in each module database **105***k*. With the use of separate module databases **105***k*, a first module database **105***k* could record data associated with a first module **101**, and a second module database could record data associated with a second module **101**. A server **105** could access each of the first module database **105***k* and the second module database **105***k*, possibly through a secure connection data transfer **802**, in order to support communication with a plurality of modules **101**.

**[0147]** User **183** can comprise an individual, business manager, network engineer, systems administrator, other employee with functional responsibilities for a system **199** (or components within a system **199** or system **100**) accessing application **171***i* using a computer with a user interface such as, but not limited to, a web browser **183***a*. Application **171***i* could also send an email or text message to user **183** if an alarm condition is detected in system **199**, such as, but not limited to, if a sensor **101***f* measurement exceeds a prescribed threshold value. The web browser **183***a* could use a connection **184** to access a web portal **171***j* operating on application **171***i*. Connection **184** could include hypertext markup language (HTML) messages, and could be through a secure connection such as, but not limited to, TLS or IPsec, although

other possibilities exist as well to those of ordinary skill in the art. Any module **101**, such as, but not limited to, module **101** A, could use the Internet **107** and establish a primary connection **181** with server **105** A, and also module **101** A could establish a backup connection **182** with server **105** B if the primary connection **181** is not available. Alternatively, any module **101**, such as, but not limited to, module **101** A, could communicate with more than one server **105** concurrently or in sequence, such that module **101** A communicates with both server A **105** and server B **105**. According to exemplary embodiments, during an active state between periods of sleep or being dormant, module **101** may communicate with more than one server **105**, such as, but not limited to, a first server A **105** and a second server B **105**. Other possibilities for a plurality of modules **101** to communicate with a plurality of servers **105** exist without departing from the scope of the present invention.

**[0148]** As contemplated herein a system **199** and other systems illustrated in additional Figures can include a set of servers **105***n*, and the exemplary system illustrated in system **199** includes at least two servers **105** in the set of servers **105***n*. Other servers besides a server **105** can be included in a set of servers **105***n*, such as, but not limited to, shared module database **105***k* which could operate on separate computers than a server **105**. Other possibilities exist as well for the number of servers **105** in a set of servers **105***n*. In another embodiment, the set of servers **105***n* could comprise a single server **105**. Thus, a set of servers **105***n* can include from one to many servers **105**.

**[0149]** FIG. **2**

**[0150]** FIG. **2** is a graphical illustration of an exemplary system, where a module sends a message to a server, and where the server responds to the message, in accordance with exemplary embodiments. Module **101** as depicted and described in FIG. **2** can operate as a wireless module **101**, although a wired connection to the Internet **107** could alternatively be utilized. System **100** as illustrated in FIG. **2** includes RF signals **201**, module IP address **202**, port number **203**, module IP:port **204**, server port number **205**, server ID **206**, server IP:port number **207**, message **208**, response **209**, wireless network firewall address **210**, and firewall port binding packet **211**. Many of the elements illustrated within system **100** in FIG. **2** are also depicted and described in connection with FIG. 2 of U.S. patent application Ser. No. 14/039,401 (the contents of which are hereby incorporated by reference in their entirety). As contemplated herein, a wireless module **101** can comprise a module **101**, or in other words a wireless module **101** may be a module **101** that is wireless. Functions described as being performed by a wireless module **101** may also be performed by a wired module **101** (where connection to a wired network would be used instead of connection to a wireless network **102**). Also as contemplated herein and illustrated in FIG. **2**, the wording "module **101** sends a message **208**" can also be considered equivalent to "server **105** receives a message **208**". Likewise, the wording "server **105** sends a response **209**" can be considered equivalent to "module **101** receives a response **209**".

**[0151]** A wireless module **101** can wake from a dormant state in order perform (i) remote and automated monitoring and (ii) control functions such as, but not limited to, collecting a sensor **101***f* measurement, communicating with server **105**, and controlling an actuator **101***y*. If module **101** is connected to land-line power or a long-lasting external power source such solar power, then module **101** may remain in an active state and bypass a dormant state, although transmitting RF signals **201** may preferably only be utilized when communicating with wireless network **102** or sending data to and receiving data from server **105**. The wireless module can acquire an IP address **202** from the wireless network **102**. IP address **202** is illustrated as being an IPv6 address, but IP address **202** could also be an IPv4 address.

**[0152]** In order to transmit or send data from wireless module **101** to server **105**, a wireless module **101** can use module program **101***i* to collect data from a sensor **101***f* in order to update server **105**. Module program **101***i* can request a port number **203** from operating system **101***h* in order to have a source IP:port for sending data using IP protocols such as, but not limited to, TCP and UDP. The terminology "IP:port" as described herein refers to combining an IP address with a port number. Wireless module IP address **202** and port number **203** can be combined to form IP:port number **204**. IP:port number **204** can be utilized as a source IP:port number for packets transmitted from wireless module **101**, as well as a destination IP:port number for packets received by wireless module **101**, when communicating with server **105**.

**[0153]** In order to utilize Internet **107**, module **101** may also need a destination IP address and port number in order to send packets to server **105**. Before sending data to server **105**, wireless module **101** preferably retrieves server IP address **106** and server port number **205** from RAM **101***e*. Server IP address **106** could be recorded in RAM **101***e* via (i) a DNS query using server name **206** or (ii) queries to M2M service provider **108** or wireless network **102**. CPU **101***b* may copy server IP address **106** and server port number **205** from nonvolatile memory into volatile memory such as, but not limited to, a register for processing to send a packet to server **105**. Server name **206** could also be a server identity. (A) Server IP address **106** or server name **206** and (B) server port number **205** could be recorded in a nonvolatile memory such as, but not limited to, flash memory **101***w* so that wireless module **101** can store the proper destination of packets transmitted or sent even when wireless module is dormant or shutdown, which avoids the processing and bandwidth requirements of obtaining server IP address **106** and server port number **205** every time the wireless module **101** wakes from the dormant or shutdown state. Server IP address **106** and server port number **205** can be combined into a server IP:port number **207**.

**[0154]** After collecting data from a sensor, module **101** can send a packet from IP:port **204** to IP:port **207**, and the packet could comprise a message **208** that may include the data from a sensor **101***f*. Note that message **208** does not need to include sensor data, and message could potentially be a periodic registration message or keep-alive message. As contemplated herein, the term "sensor measurement" can refer to data associated with or derived from a sensor **101***f*. A sensor measurement, can comprise a string containing data regarding a parameter of a monitored unit **119** and collected by a sensor **101***f*. The sensor measurement as sent in a message **208** can also represent a string (alphanumeric, binary, text, hexadecimal, etc.), where the string comprises a transformation or processing of sensor data collected by a CPU **101***b*, such including formatting, compressing, or encrypting, encoding, etc. of sensor data. A "sensor measurement" could comprise a plurality of data from a sensor **101***f*.

**[0155]** In order to minimize bandwidth and time required for RF signals **201** to be active, module **101** can send the message **208** as a single UDP datagram in accordance with a

preferred exemplary embodiment. The single UDP datagram in this embodiment can preferably be the only packet sent from module **101** to server **105** or M2M service provider **108** during a wake state for the module **101** when the radio **101***z* is active and transmitting, such as, but not limited to, in a radio resource control (RRC) connected state. In other words, according to this preferred exemplary embodiment, the message **208** sent by module **101** can preferably be the only message or packet sent by the wireless module to the server **105** between dormant periods of module **101**. By sending message **208** as a single UDP datagram, both a battery **101***k* is conserved and utilization of valuable RF spectrum is reduced. Message **208** could also comprise a series of associated UDP messages.

**[0156]** Also, as contemplated herein, message **208** could comprise a related series of packets, so that message **208** could comprise multiple datagrams. As one example, if TCP is utilized as the transport protocol for message **208**, then the series of TCP messages including the initial handshake, one or more packets of payload data, and the closing of the connection could together comprise message **208**. As another example, if UDP or UDP Lite is utilized for the transport protocol, and payload data exceeds a maximum transmission unit (MTU) size for the UDP packet and the payload data is spread across multiple packets, then the multiple packets would comprise a message **208**. Further, a related series of packets comprising a message **208** could be identified by using the same source IP:port number as either (i) received by server **105** or (ii) sent by module **101**. In addition, a related series of packets comprising a first message **208** could be identified as a series of packets sent by module **101** before receiving a response **209** from a server, and packets sent after receiving a response **209** could comprise a second message **208**. Other possibilities for a message **208** to comprise multiple packets or datagrams may exist without departing from the scope of the present invention.

**[0157]** The UDP datagram for message **208** could also be formatted according to the UDP Lite protocol, as specified in IETF RFC 3828, which is also incorporated by reference herein. The term "UDP Lite" described in the present invention may also refer to any connectionless protocol widely supported on Internet **107** where checksums may be partially disabled, thereby supporting the transfer of bit errors within a datagram. The advantages of UDP over TCP is that UDP can be quickly sent, while TCP requires a "handshake" with the server which requires more time and bandwidth, which would utilize more energy from battery **101***k*. According to an exemplary embodiment, both message **208** and response **209** can be TCP messages. In this exemplary embodiment, message **208** and response **209** could each comprise a series of TCP messages that can include a TCP SYN, SYN ACK, ACK, ACK w/data, FIN ACK, etc.

**[0158]** According to an exemplary embodiment, module **101** sends (and server **105** receives) the same sensor data in multiple copies of the same UDP packet. Each of the multiple copies of the same UDP packet can also optionally be formatted according to the UDP Lite protocol. As one example, wireless module sends three identical copies of the UDP or UDP Lite packet that include the same sensor data. The benefit of sending three copies of UDP Lite include (i) the RF signals **201** received by the base station **103** could include bit errors, which could result in a regular (RFC 768) UDP packet being dropped, since a bit error could result in a UDP checksum mismatch, as received and processed by wireless network **102**. Note that the use of checksums is mandatory in IPv6, and thus checksums cannot be fully disabled in IPv6. With UDP Lite packets transmitted by wireless module **101**, where the mandatory checksum for IPv6 can cover the packet header, wireless network **102** can forward all packets received, potentially including bit errors, to server **105** over the Internet **107**.

**[0159]** Server **105** can receive the multiple copies of the UDP or UDP Lite packets, which could include bit errors received, and server **105** could compare or combine the multiple copies or each individual UDP Lite packet in order to remove bit errors. Note that UDP Lite is not required, and wireless module **101** could send the message **208** using a single UDP packet, or multiple copies of a regular UDP (i.e. non UDP Lite) packet. However, using UDP Lite with multiple packets sent can provide benefits such as if the sensor data is encrypted in the packet, then a single bit error would normally break the receiver's ability to decipher the data using a cryptographic key, unless the encrypted data was channel coded and the channel coding could recover from the bit error in order to present an error-free input of the encrypted data to a deciphering algorithm.

**[0160]** Further, between periods of sleep when a wireless module **101** becomes active and transmits RF signals **201**, module **101**, which may also comprise a wireless module **101**, could send the sensor data in a single UDP Lite packet where the packet includes channel coding, which can also be referred to forward error correction. Forward error correction could also be implemented by sending multiple copies of the same UDP packet. Note that since large segments of message **208** could include encrypted or hashed data, those segments may not be appropriate for compression since the data is often similar to random strings which are not readily compressed. Channel coding techniques for the data in message **208** could include block codes and convolution codes. Block codes could include Reed-Solomon, Golay, BCH, Hamming, and turbo codes. According to a preferred exemplary embodiment, data within message **208** is sent as a UDP Lite packet using a turbo code to correct multiple bit errors within a packet or datagram sent by module **101** and received by server **105**.

**[0161]** In system **100** illustrated in FIG. **2**, server **105** can use IP:port **207** to receive the packet from wireless module **101** and sent from source IP:port **204** to IP:port **207**, and the packet could comprise a message **208** that may include the data from a sensor associated with module **101** or monitored unit **119**. As contemplated herein, a message **208** illustrated in FIG. **2** does not need to include sensor data and other data could be transmitted or sent, such as, but not limited to, a server instruction **414** (described in FIG. **4** below), or other data pertaining to module **101** or a monitored unit **119**. Note that server **105** can use IP:port **207** to receive a plurality of messages **208** from a plurality of wireless modules **101**. Server **105** preferably listens for UDP packets on IP:port **207** or monitors IP:port **207**, although TCP packets could be supported as well. If server **105** receives multiple copies of the same UDP packet from module **101**, server **105** preferably includes a timer to drop duplicate packets received outside a timer window such as, but not limited to, an exemplary 5 seconds.

**[0162]** After receiving the message **208** and processing the message according to the techniques described below such as, but not limited to, in FIG. **4**, server **105** can send a response **209**. Since module **101** may belong to a wireless network **102**

which includes a firewall **104**, the source IP:port of the message **208** received by server **105** could be different from the source IP:port **204** utilized by wireless module **101**. The source IP:port in message **208** could be changed if firewall **104** performs network address translation (NAT), as one example. Server **105** may not readily know if a NAT translation has been performed on the message **208**. Alternatively, firewall **104** may not perform NAT, but could still block data from the Internet **107** which does not properly match the firewall rules. As one example, firewall **104** could be a symmetric firewall (but without NAT functionality), where only packets from IP:port **207** to IP:port **204** are allowed to pass the firewall after message **208** has been sent by module **101**.

**[0163]** In either case, where firewall **104** may or may not perform NAT routing, server **105** preferably sends the response **209** from the server IP:port **207** to the source IP:port it receives in message **208**. According to a preferred exemplary embodiment, response **209** is a UDP packet sent from server **105** with (i) a source IP:port **207** and (ii) a destination IP:port equal to the source IP:port received in message **208**, as illustrated in packet **209***a*. The example use of source and destination IP:ports in message **208** and response **209** are also illustrated in FIG. **6***a* below. In this manner, the UDP packet can traverse a firewall **104**, if firewall **104** is present. If firewall **104** is present and performs NAT routing, then firewall **104** can receive the response **209** and change the destination IP address and port within response **209** to equal IP:port **204**.

**[0164]** According to exemplary preferred embodiments, module **101** may also obtain power from a land-line source, such as, but not limited to, a traditional 120 volt wall socket, or possibly power over Ethernet, and other non-transient power sources could be utilized as well. In this case, module **101** may remain persistently connected to the Internet through either a wireless network **102** or a wired connection such as, but not limited to, Ethernet. In other words, module **101** may omit entering periods of sleep or dormancy where inbound packets from the Internet would not be received due to the sleep state of module **101**. Consequently in an exemplary embodiment, module **101**, which does not sleep for periods longer than a minute, may preferably periodically send a firewall port binding packet **211** from IP:port **204** to IP:port **207** in order to keep ports and addresses within a firewall **104** and/or firewall **124** open to communications between module **101** and server **105**. Firewall port binding packet **211** can comprise a packet that is sent periodically using a timer interval that is shorter than the port-binding timeout period **117** on a firewall **104** and firewall **124**.

**[0165]** Continuing with this exemplary embodiment where module **101** does not sleep for periods longer than approximately one minute, if UDP is utilized for message **208** and response **209**, then a small UDP packet comprising firewall port binding packet **211** can be sent periodically such as, but not limited to, every 45 seconds. If TCP is utilized for message **208** and response **209**, then a small TCP packet comprising firewall port binding packet **211** can be sent periodically such as, but not limited to, every 4 minutes. Other possibilities for the timing of sending firewall port binding packet **211** are possible as well. By sending firewall port binding packet **211** periodically, server **105** can send module **101***a* response **209**, (i) which could include a module instruction **502** as explained in FIG. **5***a*, at (ii) time intervals between message **208** and response **209** that are longer than the firewall port binding timeout values **117** of firewall **104** and/or firewall **124**. Without firewall port binding packet **211**, if (A) a response **209** sent from server **105** at an exemplary 180 seconds after receiving message **208**, such as, but not limited to, after a firewall port binding timeout value **117** of firewall **104** of an exemplary 60 seconds transpired, then (B) response **209** would be dropped by firewall **104** and the response **209** would not be received by module **101**.

**[0166]** FIG. **3**

**[0167]** FIG. **3** is a flow chart illustrating exemplary steps for a server to receive a message from a module, in accordance with exemplary embodiments. As illustrated in FIG. **3**, FIG. **3** can include steps used by a module controller **105***x* in a server **105** as illustrated in FIG. **1***c*. The processes and operations, including steps for module controller **105***x*, 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.

**[0168]** 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.

**[0169]** 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.

**[0170]** 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 as limited to any one set of computer program instructions.

**[0171]** 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. 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.

**[0172]** 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.

**[0173]** At step **311**, the server **105** can record a module public key **111**, or a plurality of module keys **111** in a module database **105***k*. The module public key **111** could be received in a message **208** according to steps **516** and **517**, including authenticating the message **208**, as depicted and described in connection with FIG. **5***b* below. Module public key **111** could also be recorded at step **311** before module **101** connects to the Internet **107** the very first time, and in this case module public key **111** could be recorded in server **105** by M2M service provider **108** or module provider **109**. At step **312**, the server **105** can open a TCP/UDP socket associated with an IP:port number **207** and listen or monitor for incoming message from modules. At step **313**, server **105** can receive a message **208** sent by module **101**, using the IP:port number **207**. Although not illustrated in FIG. **3**, upon the first communication from module **101** by server **105** where the communication could include step **313**, according to an exemplary embodiment, module **101** can also send a certificate **122** to server **105**, where certificate **122** would normally include module public key **111** and module identity **110**. Server **105** could utilize a certificate **122** to verify a module identity **110**, as described in FIG. **4** below at step **412**.

**[0174]** An exemplary format of message **208** is depicted and described in connection with FIG. **6***a* below, and other possibilities for a message **208** exist as well. Although not illustrated in FIG. **3**, after receiving message **208** at step **313**, server **105** may also process any channel coding present in message **208** in order to eliminate any bit errors received. The channel coding could be included in a message **208** that utilizes the UDP Lite protocol in an exemplary embodiment. At step **314**, server **105** can decrypt a message **208** using a cryptographic algorithm **141** and one of (i) server private key **105***c*, or (ii) a symmetric key **127**. Additional details regarding step **314** are depicted and described in connection with FIG. **4** below. At step **315**, server **105** can verify that message **208** was sent by module **101** using a module identity **110**, module public key **111**, and a cryptographic algorithm **141**, including verifying a module digital signature **405** discussed in FIG. **4** below. Additional details regarding step **315** are depicted and described in connection with FIG. **4** below. Note that step **315** can take place before step **314** if the module identity **110** and/or a digital signature is not encrypted within message **208** (i.e. a sensor measurement in message **208** could be encrypted but a module identity **110** or digital signature may not be encrypted). Step **315** may optionally be omitted, if a symmetric key **127** is used to cipher data within message **208**, such that a module digital signature from module **101** was previously verified when the symmetric key **127** was implemented.

**[0175]** After verifying the identity of module **101** in step **315**, at step **316** server **105** can record sensor data or sensor measurements within message **208** in a module database **105***k*, if message **208** has a sensor measurement. Note that message **208** may not have a sensor measurement, and in this case step **316** can be skipped, or message **208** may also include other data besides a sensor measurement. Sensor data recorded in module database **105***k* can be made available for subsequent processing by server **105** or other servers or applications associated with an M2M service provider **108** in order to manage the function and operation of module **101** or monitored unit **119**. As illustrated in FIG. **7** through FIG. **9**, received sensor data could also be forwarded by server **105** to an application server **171**. Although not illustrated in FIG. **3**, in an exemplary embodiment at step **316** server **105** could alternatively forward the sensor data to application **171***i* instead of recording the data in module database **105***k*. After receiving message **208**, server **105** can process a response **209** at step **317***a*. Step **317***a* can comprise encrypting an instruction, where the instruction could include an acknowledgement of the message received, a command or setting for an actuator, and/or another control message for module **101**. Server **105** can utilize a module public key **111** and cryptographic algorithms **141** in order to encrypt the instruction. Step **317***b* can comprise creating a digital signature for the response **209** using the server private key **105***c* and cryptographic algorithms **141**.

**[0176]** Additional details regarding steps **317***a* and **317***b* are depicted and described in connection with FIG. **5***a* below. Note that step **317***a* and/or step **317***b* may optionally be omitted, such that response **209** is transmitted without encryption and/or a signature, and security could be obtained through other means, such as through firewalls **104** and **124**, or using a secured network link between module **101** and server **105**, such as, but not limited to, setting up a virtual private network (VPN) or SSH tunnel between the two endpoints. These alternative means for security at the network layer would likely require additional bandwidth and power consumption for a module **101** and thus may not be adequately efficient. As one example, if module **101** is a wireless module that sleeps for relatively long periods such as, but not limited to, every hour (and obtains a new IP address for every wake period), setting up a new VPN between module **101** and server **105** in order to receive send a message from module **101** may not be practical due to the extra drain on a battery **101***k* for re-establishing the VPN. Or, only portions of steps **317***a* and **317***b* could be used, such that a response **209** (or a message **208** received in step **313**) is not encrypted but a digital signature is used in the response **209** (or message **208**).

**[0177]** After completing steps **317***a* and **317***b*, at step **209***a*, server **105** can send response **209** from (a) the source port utilized to receive message **208** to (b) a destination IP:port. The destination IP:port can comprise the source IP:port in message **208** as received by server **105**, and the destination IP:port can represent the external interface of a firewall **104**.

In other words, server **105** may send response **209** from server IP:port **207** to the source IP:port received in message **208**, which could represent the source IP:port on a wireless network firewall **104**, wherein the source IP:port on the wireless network firewall **104** contains the firewall IP address **210**. The wireless network firewall **104** could forward the response **209** to module IP:port **204**. As contemplated herein, server **105** can send response **209** as soon as practical after receiving message **208**, and in any case response **209** should be sent before the expiration of a firewall port binding timeout value **117** associated with firewall **104**. According to a preferred exemplary embodiment, response **209** is sent by server **105** within no more than 5 seconds of receiving message **208**. After completing step **209***a* as illustrated in FIG. **3***b*, server **105** can return to step **312** and listen for or monitor for additional incoming messages **208** from modules **101**.

**[0178]** FIG. **4**

**[0179]** FIG. **4** is a flow chart illustrating exemplary steps for a server to process a message, including verifying a module's identity and decrypting data, in accordance with exemplary embodiments. The steps illustrated in FIG. **4** may comprise step **315** and step **316** illustrated in FIG. **3** above. Server **105** can receive message **208** using IP:port **207**, as illustrated in FIG. **2**. Message **208** can be formatted according to the UDP protocol or UDP Lite protocol, although other possibilities exist as well without departing from the scope of the present invention

**[0180]** At step **407**, server **105** can process the packet using the appropriate transport layer protocol, such as, but not limited to, UDP. In this step **407**, the body of the packet comprising message **208** can be extracted, and a checksum, if any, can be calculated to verify the integrity. Note that if the UDP Lite protocol is utilized, the checksum may optionally only apply to the packet header. At step **408**, server **105** can remove channel coding, if present in message **208**. Channel coding techniques utilized in step **408** could include block codes and convolution codes, and can use the same channel coding algorithms used in channel coding algorithms implemented by module **101**, depicted and described in connection with FIG. **5***a* below. By processing channel coding in step **408**, server **105** can correct potential bit errors received in message **208**, although channel coding **408** may be optionally omitted. As noted above, the use of channel coding **408** can be preferred in an embodiment, since any bit errors received within module encrypted data **403** in message **208** could break (i) a cryptographic algorithms **141** used by server **105** at subsequent steps **413**, and/or (ii) the verification of module digital signature **405** at step **410** below.

**[0181]** At step **409**, the server **105** can read and record the module identity **110**, if module **110** is included in message **208** as external to module encrypted data **403** as illustrated in an exemplary message **208** in FIG. **6***a* below. Although not illustrated in FIG. **4**, server **105** can select a module public key **111** for module **101** by querying a module database **105***k* with module identity **110**. Module identity **110** could comprise a string or session identifier, whereby server **105** could derive or track a module identity **110** from one message **208** to the next message **208** using the string or session identifier. By including module identity **110** in a message **208**, but external to module encrypted data **403** such as, but not limited to, illustrated in FIG. **6***a* below, a server **105** can utilize module identity **110** in order to select a server private key **105***c* or symmetric key **127** for decrypting module encrypted data **403**. According to an exemplary embodiment, a plurality of server private keys **105***c* could be utilized, where a first private key **105***c* is used with a first set of modules **101** and a second private key **105***c* is used with a second set of modules **101**. The first and second private keys **105***c* could use or be associated with different sets of parameters **126**. By reading the module identity **110** outside of module encrypted data **403**, the module identity **110** can be read before decryption, in order to identify which of the first or second set server private keys **105***c* that a module **101** sending message **208** is associated with, and thus server **105** can subsequently select the first or second set of server private keys **105***c* to use when decrypting module encrypted data **403**.

**[0182]** Alternatively according to an exemplary embodiment, if server **105** operates in a distributed environment (such as, but not limited to, comprising multiple sub-servers **105***w* as illustrated in FIG. **10**, an unencrypted module identity **110**, including a possibly a session identifier for module identity **110** within a message **208**, can be utilized by a message preprocessor **105***y* to select the appropriate sub-server **105***w* to process the message **208**. Server **105** using message preprocessor **105***y* could forward the message **208** to the correct sub-server **105***w*. At step **410**, server **105** can validate and verify the module identity **110** using the module digital signature **405** inserted by module **101** in message **208**. Module digital signature **405** can comprise a secure hash signature or tag, where module **101** generated the hash signature using the module private key **112** and digital signature algorithms **141***d*. As one example, server **105** can utilize the module public key **111** recorded in memory **105***e* to securely validate the module digital signature **405** receive in a message **208**.

**[0183]** The module digital signature **405** can be verified according to public key infrastructure (PKI) standards such as, but not limited to, the National Institute of Standards (NIST) "FIPS 186-4: Digital Signature Standard", or IETF RFC 6979 titled "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)". Other PKI standards or proprietary techniques for securely verifying a module digital signature **405** may be utilized as well. If message **208** comprises an initial communication from module **101**, at step **412** server **105** can verify that module public key **111** is associated with module identity **110** using a module certificate **122**, where certificate **122** includes a signature **123** from a certificate authority **118**, as illustrated in FIG. **1***h* of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix. Server **105** could receive certificate **122** before module **101** sends message **208**, or server **105** could query module **101** or another server for certificate **122** after receiving message **208**. Server **105** could use digital signature algorithms **141***d* to compare a secure hash calculated using (i) a first certificate **122** and/or public key from module **101** and (ii) a second certificate and/or public key from certificate authority **118** or another server, in order to confirm that module public key **111** is associated with module identity **110**, where module identity **110** was read from message **208** in step **409**. The secure hash could also be calculated using module public key **111** and a public key from certificate authority **118**, and other possibilities using PKI exist as well for server **105** to confirm module public key **111** is associated with module identity **110** at step **412**.

**[0184]** In an exemplary embodiment, if (A) module encrypted data **403** includes module identity **110** and/or module digital signature **405**, then (B) steps **409** and/or **410** may also take place after step **413**, where server **105** (*i*) first

decrypts module encrypted data **403** and can then (ii) verify module identity **110** by performing steps **409** and **410** after step **413**. If module encrypted data **403** utilizes a symmetric cipher **141***b*, then a module identity **110** can preferably be external to module encrypted data **403** so that server **105** can select the appropriate symmetric key **127** used by module **101** in order to decipher module encrypted data **403** (since a plurality of modules **101** may communicate with server **105** concurrently).

**[0185]** After verifying module digital signature **405** in step **410**, server **105** can record an authenticated module encrypted data **403** from module **101** received in message **208**. At step **413**, server **105** can decrypt module encrypted data **403** using cryptographic algorithms **141** and either (i) server private key **105***c* as a decryption key with asymmetric ciphering **141***a* or (ii) symmetric key **127** with symmetric ciphering **141***b*. A symmetric key **127** may be stored in a module database **105***k*, as noted in FIG. **1***f* above. If a symmetric key **127** is used at step **413**, the symmetric key **127** could be (i) sent by server **105** in a response **209** or (ii) received by server **105** in a prior message **208**, before the message **208** illustrated in FIG. **4** was received by server **105**.

**[0186]** With an asymmetric ciphering **141***a* scheme used in a module encrypted data **403** and by cryptographic algorithms **141** at step **413**, server **105** can decrypt module encrypted data **403** using (i) server private key **105***c* and (ii) RSA algorithms **153**, elliptic curve cryptography (ECC) algorithms **154**, or other algorithms for public key cryptography. The use and application of RSA algorithms **153** and cryptography are described within IETF RFC 3447, among other published standards. The use and application of ECC cryptography and algorithms are described within IETF RFC 6637, among other published standards. ECC algorithms **154** may be preferred in order to maintain high security with smaller key lengths, compared to RSA, in order to minimize the message lengths, radio frequency spectrum utilization, and processing power or energy required by module **101**. Note that module encrypted data **403** may also include a security token **401** (not shown in FIG. **4**, but shown in FIG. **5***a*), which could comprise a random string, and thus each module encrypted data **403** received by server **105** in message **208** may be reasonably considered unique and thus robust against replay attacks.

**[0187]** With a symmetric ciphering **141***b* scheme used in a module encrypted data **403** and by cryptographic algorithms **141** at step **413**, server **105** can decrypt module encrypted data **403** using (i) symmetric key **127** and (ii) a symmetric cipher **141***b* such as, but not limited to, AES **155**, Triple DES, or similar secure symmetric ciphers. As one example, by using ECC cryptography and ECIES, server **105** could decrypt module encrypted data at step **413** by using the steps outlined in FIG. **3**, titled "ECIES Encryption Functional Diagram" in "A Survey of the Elliptic Curve Integrated Encryption Scheme" by Martinez et al in the Journal of Computer Science and Engineering, Volume 2, August 2010, page 11, (herein incorporated by reference). Other possibilities exist as well without departing from the scope of the present invention. Server **105** can utilize step **413** illustrated in FIG. **4** to extract the plaintext, or decrypted data within module encrypted data **403**.

**[0188]** After decrypting module encrypted data **403**, server **105** can read the resulting data within message **208**, which could comprise a server instruction **414**. The server instruction **414** can represent the purpose of the message **208** for server **105**. Server instruction **414** could comprise a plurality of different procedures for server **105**, such as, but not limited to, an "update" with sensor data, a "query" for data or instructions from server **105** or M2M service provide **108**, a "notification" of state or condition at module **101** such as, but not limited to, an alarm or error, a "configuration request" where module **101** seeks configuration parameters, a "software request" where module **101** request updated software or routines, a "registration" message where module **101** periodically registers with server **105**, etc. Thus, server instruction **414** can comprise the purpose module **101** sends message **208**. In addition, server instruction **414** could comprise a "confirmation", where module **101** sends a "confirmation" in a second message **208** after receipt of a response **209**, where response **209** could include a module instruction **502** (below), and the "confirmation" in this second message **208** could signal server **105** that the module instruction **502** had been properly executed.

**[0189]** As examples for server instruction **414**, an "update" could be used to periodically notify server **105** of regular, periodic sensor data **305** acquired by a sensor **101***f*. An "update" for server instruction **414** may also comprise a periodic report regarding monitored unit **119** or information regarding a state, condition, or level for an actuator **101***y*. A "query" for server instruction **414** could comprise module **101** querying server **105** for data from a module database **105***k*, where the data could be associated with monitored unit **119**, wireless network **102**, an element within module **101** such as, but not limited to, an actuator setting. A "notification" for server instruction **414** could comprise module **101** notifying server **105** that an alarm or error condition has occurred, such as, but not limited to, a sensor measurement exceeds a threshold value or another error condition such as, but not limited to, loss of contact with monitored unit **119**. A "configuration request" for server instruction **414** could comprise module **101** requesting server **105** for configuration parameters or a configuration file. Other possibilities for server instruction **414** exist without departing from the scope of the present invention.

**[0190]** At step **415**, server **105** can process the server instruction **414**. If server instruction **414** comprises an "update", then sensor data, or other data in server instruction **414** including potentially a new symmetric key **127** generated by module **101**, could be recorded in module database **105***k*, Other applications may subsequently access the sensor data for generating reports or making decisions regarding monitored unit **119**. If server instruction **414** comprises a "query", then server **105** could execute the query at step **415**. If server instruction **414** comprises a "notification" of an alarm, then step **415** could initiate procedures for alarm notification to 3$^{rd}$ parties or alarm resolution. Other possibilities for processing a server instruction **414** at step **415** exist without departing from the scope of the present invention.

**[0191]** FIG. **5***a*

**[0192]** FIG. **5***a* is a flow chart illustrating exemplary steps for a server to process a response for a module, including sending and signing a module instruction, in accordance with exemplary embodiments. The steps illustrated in FIG. **5***a* may comprise step **317***a* and step **317***b* illustrated in FIG. **3** above. Since message **208** and response **209** may traverse the public Internet **107**, a module **101** and a server **105** may prefer to take additional steps to sending plaintext in packets in order to maintain security of a system **100**. Server **105** can process a response **209** to a message **208** from module **101** using a

module public key **111** and a server private key **105***c*, according to a preferred exemplary embodiment. If a symmetric cipher **141***b* is utilized within cryptographic algorithms **141**, then server **105** may also utilize a symmetric key **127** to encrypt data within a response **209**. Note that the security methods described herein are optional, and message **208** and response **208** can be sent without any or all of the additional security steps described herein, but the use of these security steps may be preferred.

**[0193]** After receiving message **208** as illustrated in FIG. **2**, server **105** can prepare an acknowledgement **501**. The acknowledgement **501** can be a simple text, binary, or hexadecimal string to confirm that message **208** has been received and/or processed by server **105**. Since message **208** may be transmitted via a UDP or UDP Lite packet, module **101** may preferably utilize a reply message from server **105** containing acknowledgement **501**, in order to confirm message **208** has been received by server **105**. Alternatively, if TCP is used to transmit message **208**, an acknowledgement **501** may be used at the application layer of the Open Systems Interconnection (OSI) model, wherein a simple TCP ACK message may operate at the lower transport layer than the application layer. UDP may be preferred over TCP in order to reduce processing resources for module **101** and server **105**, especially considering the relatively small and comparably infrequent messages sent between a module **101** and a server **105** (when compared to web browsing and considering module **101** may have a battery **101***k* that may preferably last for weeks or longer without recharging). In processing a response **209**, server **105** may optionally add a security token **401**, which could be a random number **128***a*, or a randomly generated text, binary, or hexadecimal string. Security token **401** could be a random number **128***a* or string that is included in response **209** in order to make each response **209** unique and thus avoid any replay attacks when response **209** traverses Internet **107**. Note that a message **208** may also preferably include a security token **401**.

**[0194]** In other words, the use of security token **401** can ensure to a high level of certainty that each response **209** will be different and thus the data within response **209** would not be sent more than once. Note that security token **401** may be generated by module **101** in message **208**, and in this case server **105** can use the same security token received in message **208**. Security token **401** can alternatively be generated by server **105** and different than any security token **401** received in message **208**. Security token **401** illustrated in FIG. **5***a* can be derived or processed by using message **208** in accordance with preferred exemplary embodiments.

**[0195]** Server **105** may also optionally add a module instruction **502** when preparing a response **209**. The module instruction **502** could be a string that contains instructions or configuration parameters for module **101**, such as, but not limited to, an order to change state, parameters regarding the monitoring of monitored unit **119**, server names or addresses, radio frequency parameters, wireless network **102** authentication parameters or keys, keys for communication with server **105** or M2M service provider **108**, etc. Module instruction **502** may also comprise an instruction to change the state of actuator **101***y*, a timer value, a sensor threshold value, the threshold for an alarm state, and information for display at a user interface **101***j*, an instruction to sleep, etc. Module instruction **502** may further comprise an updated module private key **112**, and updated server public key **114**, or the address or name of a new server **105** added to M2M service provider **108**. According to an exemplary preferred embodiment, a module instruction **502** could comprise a "key generation" instruction, where module **101** generates a new pair of a module private key **112** and a module public key **111**, utilizing the exemplary steps and procedures illustrated in FIG. **5***b* below, including step **515**. The "key generation" **608** module instruction **502** (illustrated in FIG. **6***a* below) could be used to create new keys for a new purpose (such as, but not limited to, connecting to a new wireless network **102** or communicating with a new server **105**), while the existing keys used to communicate with server **105** could remain operable or be deprecated at a later time. Alternatively, an existing or previous module public key **111** could be deprecated or become invalid once server **105** sends a "key generation" module instruction **502**.

**[0196]** In order to control module **101**, server **105** would normally need to include module instruction **502** in the response **209** only after receiving message **208**, since the server **105** would normally not be able to send messages to a module **101** at arbitrary times, such as before a message **208** has been received by the server **105**. The reasons include (i) the module may normally be in a sleep or dormant state, in order to conserve battery life or power consumption, where an unsolicited incoming Internet packet from server **105** would not be received by module **101**, and (ii) a wireless network **102** (or equivalent wired network that a wired module **101** could connect with) may frequently include a firewall **104**. Firewall **104** could prevent packets from the Internet **107** from reaching module **101** unless module **101** had previously first sent a packet to server **105** within a firewall port-binding timeout period **117** of firewall **104**. The port-binding timeout period of a firewall **104** may be an exemplary period such as, but not limited to, 20-60 seconds for UDP packets and several minutes for TCP packets. Note that module instruction **502** may optionally be omitted, such that (b) some response **209** messages may include module instruction **502**, and (b) other response **209** messages may omit module instruction **502**, but include an acknowledgement **501** to message **208**. Also note that according to an exemplary embodiment described herein, the use of optional strings or steps can be depicted in FIGS. **4** and **5***a* through the use of dashed lines for the various elements illustrated. In other words, the use of dashed lines shows steps that are included in one exemplary embodiment, but excluded or omitted in another exemplary embodiment.

**[0197]** Server **105** may then use as input the acknowledgement **501**, security token **401**, and module instruction **502**, including optional data and cryptographic parameters **126**, into cryptographic algorithms **141** at step **503**. The cryptographic algorithms **141** at step **503** can utilize either (i) module public key **111** as an encryption key if asymmetric ciphering **141***a* is utilized, or (ii) a shared symmetric key **127** if a symmetric cipher **141***b* is utilized, such as, but not limited to, AES **155** ciphering. The output of cryptographic algorithms **141** at step **503**, using acknowledgement **501**, security token **401**, and module instruction **502**, plus optional data and parameters **126**, as input, can be server encrypted data **504**, as illustrated in FIG. **5***a*. Server encrypted data **504** could be a string or number, including a text, binary, or hexadecimal string or series of numbers or bits, and other possibilities for the formal of server encrypted data **504** exist as well, including a file, without departing from the scope of the present invention. By using module public key **111** and/or symmetric key **127** in the cryptographic algorithms **141** at step **503**, server encrypted data **504** may only be reasonably decrypted

by module **101** using module private key **112** and/or symmetric key **127**. Thus the response **209** transmitted across an Internet **107** may be reasonably considered secure and only reasonably decrypted by module **101**.

**[0198]** Server **105** can then process server encrypted data **504** by appending or including server identity **206**. Note that server identity **206** can be appended or included after the operation of step **503**, since the server identity **206** may optionally be openly readable within a response **209** transmitted or sent to module **101**. As one example, server identity **206** could comprise IP address **106** as a source IP address in response **209**, which would be openly readable on the Internet **107** since a valid packet must have a source and destination IP address. Additional details on an exemplary structure of response **209** are illustrated in FIG. **6***a* below. By including server identity **206** after encryption at step **503**, the module can read the server identity **206** and verify a digital signature within response **209** without having to first decrypt data within response **209** using the module private key **112** or symmetric key **127**. Note that server identity **206** could alternatively be included within server encrypted data **504**, such that step **505** takes place before step **504**. In other words, including server identity **206** external to a server encrypted data **504** can be used by module **101** to select the proper server public key **114** when verifying a digital signature in response **209**.

**[0199]** Server **105** can then process a server digital signature **506** using the server private key **105***c*. In an exemplary embodiment, the server digital signature **506** can be processed according to public key infrastructure (PKI) standards such as, but not limited to, the National Institute of Standards (NIST) "FIPS 186-4: Digital Signature Standard" (which is hereby incorporated herein by reference), or IETF RFC 6979 titled "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)" (which is hereby incorporated herein by reference). In another exemplary embodiment the use of a server digital signature **506** can be processed according to the description of a digital signature according to the Wikipedia entry for "Digital Signature" as of Sep. 9, 2013, which is incorporated by reference herein in its entirety. Also note that other uses of a digital signature as contemplated within the present invention may refer to the above three references and related standard techniques for processing and creating digital signatures. Other PKI methods for securely generating a server digital signature **506** may be utilized as well.

**[0200]** According to a preferred exemplary embodiment, ECC algorithms for generating server digital signature **506** may be utilized in order to minimize the key length compared to RSA algorithms. Server digital signature **506** may comprise a secure hash signature using a hash algorithm such as, but not limited to, secure hash algorithm 1 (SHA-1), or subsequent standards such as, but not limited to, SHA-2 and SHA-3, and other possibilities exist as well. Server digital signature **506** is illustrated in FIG. **5***a* as being processed after server encrypted data **504**, but server digital signature **506** may also optionally be included in server encrypted data **504**. Step **506** may also take place before step **505**.

**[0201]** Also note that server digital signature **506** may preferably be included in a response **209** before module **101** begins either (i) utilizing a symmetric key **127** shown in step **413** to encrypt a module encrypted data **403**, or (ii) accept or process a module instruction **502**. After including server digital signature **506** in a first response **209** that uses asymmetric ciphering **141***a*, server **105** may omit server digital signature **506** in a second subsequent response. The second subsequent response could be a case where (i) server encrypted data **504** utilizes a symmetric key **127** for ciphering (where server **105** received the symmetric key **127** in a message **208** that utilized asymmetric ciphering **141***a* as illustrated in FIG. **4** above) and (ii) expiration time **133** of symmetric key **127** has not transpired.

**[0202]** FIG. **5***b*

**[0203]** FIG. **5***b* is a flow chart illustrating exemplary steps for a server to communicate with a module that has derived a public key and private key, in accordance with exemplary embodiments. In order to utilize communications secured with PKI techniques such as, but not limited to, private keys, public keys, certificates, and identities, a module **101** may preferably obtain or generate these keys and utilize a module identity **110** and/or a certificate **122** in a secure manner. Given that a plurality of modules **101** may be deployed in potentially remote places, without frequent contact with end users or technicians, the use of secure PKI techniques for a module **101** can create a significant set of challenges for the generation of module public key **111** and module private key **112**, as well as properly and securely obtaining a certificate **122** with an module identity **110**. Using conventional technology, significant challenges and costs can be incurred when (i) module **101** has already been deployed, such as collecting data from a monitored unit **119**, and (ii) module **101** needs to utilize a new set of module private key **112** and module public key **111**.

**[0204]** Exemplary embodiments that include derivation or processing of a new module private key **112** and module public key **111** may utilize the particular steps and procedures contemplated herein, in order to minimize any potential human intervention (with related costs) while continuing to maintain or also enhance security, compared either (i) externally generating module private key **112**, and/or (ii) continuing to use the same module private key **112** for the lifetime of module **101**. Over a long period of operating time for a module **101**, such as, but not limited to, several years or longer, there may be many reasons module **101** may need a new pair of PKI keys, such as, but not limited to, (i) expiration of a certificate **122**, or the certificate **122** of a parent signature authority, (ii) the transfer of ownership or control of module **101**, where the prior ownership could have direct or indirect access to the module private key **112**, (iii) supporting a new server **105** that has different security requirements or a different set of cryptographic parameters **126** (longer keys, different ECC curves, different cryptographic algorithms **141**, etc.), and/or (iv) revocation of a public key in a chain of signatures associated with a certificate **122**. In the case of (ii) above, new ownership of module **101** may require a module **101** to utilize a new module private key **112** since the old ownership may have access to an old module private key **122**. In the case of (iii) above, a new server **105** may require a pair of public/private keys incompatible with a prior set of public/private keys utilized by module **101** and/or a certificate **122** for module **101**.

**[0205]** Other possibilities exist as well for reasons why a module **101** and/or server **105** may prefer for a module **101** to utilize a new module public key **111** and new module private key **112**. In an exemplary embodiment, module **101** may generate a new public/private key periodically in order to enhance the security of a system **100**. A benefit of a system **100** supporting periodic generation of keys by module **101** is that the key length can be shortened in order to obtain a

similar level of security, and the processing power and energy consumption, possibly from a battery **105***k*, can be reduced through the use of shorter key lengths. In other words, over time such as, but not limited to, several months or years, the use of a plurality of different pairs of public/private keys for module **101** with shorter key lengths can be both more secure and energy efficient than using a single pair of public/private keys with a longer key length for the lifetime of module **101**. Shorter key lengths may also be more compatible with processing power constraints of a module **101**. Thus, in exemplary embodiments, module **101** and/or server **105** may prefer for module **101** to periodically generate new public and private keys.

**[0206]** The general approach adopted by most mobile phone networks over the past two decades has been founded upon the use of a pre-shared secret key recorded in subscriber identity module (SIM) or UICC cards, such as the Ki pre-shared secret key in 2G, 3G, and subsequent networks. That approach may work for mobile phones, where the SIMs can often be easily replaced, but the use of a pre-shared secret key in a SIM or UICC may not be suitable for a module **101** and M2M service provider **108** for many circumstances. As one example, significant costs may be incurred by swapping out a SIM card for already deployed modules **101**, especially if they are in remote locations or continually moving such as, but not limited to, a tracking device on a container, pallet, truck, or automobile. In an exemplary embodiment, a module **101** may preferably record multiple pairs of public/private keys **111**/**112** for various and different functions, such as, but not limited to, connecting to different servers **105**, connecting to different wireless networks **102**, etc. As contemplated herein, recording more than one public/private key **111**/**112** can comprise module **101** recording a plurality of pairs of module public keys **111** and module private keys **112**. In exemplary embodiments, one pair comprising a first module public key **111** and a first module private key **112** can be identified or selected from a different pair comprising a second module public key **111** and a second module private key **112** using a module public key identity **111***a*.

**[0207]** The number of pairs of public/private keys useful to a module **101** concurrently could be several, such as, but not limited to, an exemplary three or more actively used public/private keys, although other possibilities exist as well. Manually trying to change or add a new SIM card each time a new security key is required may not be efficient or feasible. Or in another exemplary embodiment, the multiple pairs of private and public keys could be used in sequence, such that module **101** with server **105** utilizes a single module public key **111** and module private key **112** at any given point in time. In the case where module **101** with a module identity **110** derives or generates more than one module private key **112** and module public key **111** during the lifetime of module **101** and sends the derived module public keys **111** over time to a set of servers **105***n*, this case may be considered a server **105** receiving a series of module public keys for a module identity **110**. The various pairs in the series may also use either different sets of cryptographic parameters **126** or the same set of cryptographic parameters **126**. The series of module public keys **111** (with corresponding module private keys **112**) can be processed by a CPU **101***b* with key pair generation algorithms **141***e* and a random number generator **128**. The random number generator **128** can use input from a sensor **101***f*, a radio **101***z*, and/or a temporary random seed file **139**.

**[0208]** In exemplary embodiments, module **101** can use a module public key **111** for sending a module encrypted data **403** or receiving a server encrypted data **504** by either (i) sending the module public key **111** to a server **105** in order to allow the module encrypted data **403** to be decrypted (such as, but not limited to, using a step **413**) or the server encrypted data **504** to be encrypted (such as, but not limited to, using a step **503**), or (ii) inputting the module public key **111** into a key derivation function **141***f* in order to derive or process a derived shared secret key **129***b*, which could be used with a symmetric key **127**. Other possibilities exist as well for module **101** to use its own module public key **111** with cryptographic algorithms for communicating with a server **105**.

**[0209]** FIG. **5***b* illustrates exemplary steps that can be performed with module **101**, including using a module program **101***i*, for generating, deriving, and/or updating a module public key **111** and module private key **112**. The steps illustrated in FIG. **5***b* include both (i) an "initial" or "startup" case where module **101** has not previously derived keys (or keys not internally derived may not have been loaded), and (ii) a subsequent or "follow on" time where module **101** can generate or derive keys after keys were initially obtained or derived. Note that efficient and secure methods and systems contemplated herein, including in FIG. **5***b*, may also be utilized with a regular consumer mobile phone, or smartphone, as a module **101**. Mobile phones as module **101** can benefit from (i) deriving a module public key **111** and a module private key **112**, (ii) sending module encrypted data **403** in a message **208** using the derived keys, and (iii) receiving a server encrypted data **504** in a response **209** also using the derived keys. In the exemplary embodiment where module **101** comprises a mobile phone, then sensor **101***f* may comprise a microphone and actuator **101***y* may comprise a speaker, and other possibilities exist as well to those of ordinary skill in the art for module **101** to comprise a mobile phone.

**[0210]** At step **511**, during manufacturing of module **101**, including manufacturing of sub-components such as, but not limited to, a circuit board, assembly of hardware components illustrated in FIG. **1***b*, etc., a module identity **110** could be written into the hardware, and could comprise a serial number, International Mobile Equipment Identity (IMEI) number, Ethernet MAC address, or a similar persistent identification for a module **101**. An IEMI number may be used with a mobile phone as module **101**, in a preferred embodiment. For security purposes, the module identity **110** may preferably be written into a read-only location or protected location or protected memory, such as, but not limited to, a readable location on a system bus **101***d*, which could also comprise a ROM **101***c*. Recording and utilizing module identity **110** is also depicted and described in connection with FIG. **1***e*, FIG. **2**, and elsewhere herein. Alternatively, module identity **110** could be recorded in a non-volatile memory such as, but not limited to, a flash memory **101***w*.

**[0211]** At step **512**, module **101** can be distributed to end users and also installed with a monitored unit **119**. If module **101** is a mobile phone, then monitored unit **119** could be a person that carries the mobile phone. Also note that a monitored unit **119** could be omitted, and a module **101** could use the techniques contemplated herein. At step **513**, a shared secret key **510**, parameters **126**, and a server address **207** can be recorded in a nonvolatile memory **101***w*. Parameters **126** may comprise settings for a cryptographic algorithms **141** as illustrated in FIG. **1** *g*, including (i) key lengths, (ii) algorithms to utilize for key generation or ciphering, such as, but

not limited to, selecting RSA algorithms **153** or ECC algorithms **154**, (iii) a specific secure hash algorithm **141***c* to utilize, such as, but not limited to, SHA-256 or SHA-3, (iv) an expiration date of the module public key **111**, (v) a maximum time value for an expiration time **133** associated with a symmetric key **127**, (vi) a ECC parameters **137** or an ECC standard curve **138** as parameters **126** in FIG. **1***h* of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix, (vii) the specification of or values for a padding scheme for use with a digital signature algorithms **141***d*, and/or similar or related values for using cryptographic algorithms **141***d*. Although not illustrated in FIG. **5***b*, at step **512** a configuration file could also be loaded into non-volatile memory, where the configuration file includes a plurality of fields specifying the operation of module **101**. The shared secret key **510**, parameters **126**, and server address **207** could be included in a configuration file.

**[0212]** Continuing at step **513**, server identity **206** could be utilized in place of or in addition to server address **207**, and in this case module **101** can later perform a DNS or DNSSEC lookup using server identity **206** in order to obtain server address **207** for use in a message **208**, such as the destination address. Shared secret key **510** and server address **207** (or server identity **206**) could also be recorded in a ROM **101***c* at step **513**. Step **513** may also be performed concurrently with step **511** or step **512**. According to an exemplary embodiment, a manufacturer may perform step **513** and in this case step **513** could take place concurrently with step **511**. In another embodiment, a distributor of module **101** could perform step **513** and in this case step **513** could take place concurrently with step **512**. Alternatively, step **513** may be performed by a technician or end user after manufacturing and distribution and before module **101** begins collecting sensor data with a monitored unit. Other possibilities exist as well for the sequence of steps **511** through **513** illustrated in FIG. **5***b* without departing from the scope of the present invention.

**[0213]** Note that step **513** may take place multiple times during the lifetime of a module **101**, and in this case (a) the first time step **513** is conducted, step **513** could be conducted concurrent with steps **511** or **512**, and (b) a subsequent time step **513** is conducted, step **513** could be conducted after the receipt of a response **209**, where the response **209** includes a second shared secret key **510**, server address **207**, and also potentially a new module identity **110**. In other words, although not illustrated in FIG. **5***b*, a module **101** could return to step **513** from later steps upon the equivalent of a "factory reset", or similar command where flash memory **101***w* and other nonvolatile memory would be cleared. In an exemplary embodiment where step **513** takes place a second time may potentially be the transfer of ownership or control of module **101**, or a another embodiment where step **513** takes place a second time could be the upload of new firmware that is incompatible with a previous configuration file. In any case, shared secret key **510** can preferably be uniquely associated with module **101** (i.e. any given shared secret key **510** may belong only to an individual module **101**).

**[0214]** Shared secret key **510** may comprise a pre-shared secret key **129***a*, as described in FIG. **1***e*. If module **101** has already derived a module private key **112** and module public key **111** (such as when step **513** is being conducted at a second or additional time as contemplated in the previous paragraph), then shared secret key **510** may comprise (i) a key received in a server encrypted data **504** including possibly a symmetric key **127**, or (ii) a derived shared secret key **129***b*. Derived shared secret key **129***b* could be obtained from using a key derivation function **141***f* and module public key **111** and server public key **114**, using a module public key **111** that has already been derived or used by module **101** (such as if at least one module private key **112** and module public key **111** had already been used or derived before step **513**).

**[0215]** As contemplated herein in an exemplary embodiment, an first module private key **112** and first module public key **111** could be derived outside module **101** and loaded into a nonvolatile memory such as flash memory **101***w* at a prior time before step **513**, and the shared secret key **510** could be received by module **101** using the first module private key **112** and module public key **111** (such as, but not limited to, receiving the shared secret key **510** in a server encrypted data **504** using the first module private key **112** which had been loaded). Step **513** could then comprise a later time after the server encrypted data **504** has been received that includes the shared secret key **510**, where module **101** may (i) prefer to begin utilizing keys that module **101** internally derives using cryptographic algorithms **141** instead of (ii) continuing to use the first module public key **111** and module private key **112** that were derived outside of the module **101**, such as, but not limited to, possibly loaded into a nonvolatile memory from an external source.

**[0216]** In the embodiment where shared secret key **510** has not been received by module **101** in a server encrypted data **504**, shared secret key **510** could be obtained and loaded by a distributor, installer, or end user into a nonvolatile memory such as, but not limited to, flash memory **101***w* in the form of a pre-shared secret key **129***a*, where pre-shared secret key **129***a* was obtained using a module identity **110** and pre-shared secret key code **134** as depicted and described in connection with FIG. **1***e* of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix. Module **101** could also utilize a first pre-shared secret key **129***a*, including a first pre-shared secret key **129***a* entered by potentially a distributor, installer, or end-user described in FIG. **1***e*, to derive shared secret key **510**. Other possibilities exist as well for shared secret key **510**, and shared secret key **510** can be useful for the proper identification and/or authentication of module **101** upon module **101**'s generation of a private key **112** and public key **111**, as described below including step **517**. If module **101** is a mobile phone, as contemplated herein, shared secret key **510** could be loaded by a distributor or company selling or servicing the mobile phone, or shared secret key **510** could be obtained by the end user or subscriber accessing a web page associated with a mobile operator for a wireless network **102** associated with the mobile phone and/or SIM card.

**[0217]** Also note that as contemplated herein, an initial module private key **112** and initial module public key **111** could be recorded into nonvolatile memory at step **513**. For example, a manufacturer, distributor, installer, technician, or end-user could load the initial module private key and initial module public key **111**, where the initial module public key **111** would be utilized to authenticate at step **517** a subsequent set of public/private keys derived by module **101** at step **515**. In this case, the initial module public key **111** and/or initial module private key **112** described in the previous two sentences could comprise the shared secret key **510**. In another embodiment, the initial module public key **111** and initial module private key **112** could be recorded in a SIM or UICC, and the SIM or UICC could be either virtual or physical such as, but not limited to, a SIM card, including a Universal

Integrated Circuit Card (UICC) or an embedded UICC (eUICC). A set of servers **105***n* could also record the initial module public key **111** recorded in the SIM (including an eUICC), and the set of servers **105***n* could authenticate a message or a subsequent module public key **111** derived by module **101** (such as in a step **515** below) using the initial module public key **111**.

**[0218]** The use of an initial module public key **111** and/or initial module private key **112** are also depicted and described in connection with FIG. **5***b* of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix, which is hereby incorporated by reference in its entirety. Thus, FIG. **5***b* also contemplates an embodiment where shared secret key **510** at step **513** comprises an initial public/private key pair for module **101** that is not internally derived by module **101**, including keys derived at step **515**. Note that the contemplation of the use of shared secret key **510** as a pre-shared secret key **129***a* within the present invention may be different than the use of a pre-shared secret key within a subscriber identity module (SIM) card as commonly supported by wireless networks **102** with mobile phones in 2013.

**[0219]** If either a "virtual" SIM or a physical SIM card or eUICC is present within module **101** (including a UICC or eUICC), and the SIM contains a pre-shared secret key, such as, but not limited to, Ki, then as contemplated herein, shared secret key **510** may be derived using the SIM and Ki. As one example, module **101** could (i) utilize a RAND message, potentially received from a 3G or 4G mobile network such as wireless network **102**, and (ii) input the RAND into the SIM card and receive a response RES (or SRES), and utilize the string in RES to process or derive a shared secret key **510**. Response RES could also comprise a shared secret key **510**. Server **105** could also submit the same RAND associated with the SIM and Ki to wireless network **102**, and receive the same RES as obtained by module **101**. By both module **101** and server **105** having the same RES value, they can follow a pre-agreed series of steps to use the same RES in order to derive a commonly shared secret key **510** (or the shared RES could comprise a shared secret key **510**). In one embodiment where module **101** includes a SIM for a wireless network **102**, such as, but not limited to, a 4G LTE network, module **101** and server **105** could both utilize a key derivation function **141***f*, using the same RES as input, in order to derive the same shared secret key **510**.

**[0220]** At step **514**, module **101** can read module identity **110** using a read-only address. Module **101** can read module identity **110** directly from read-only hardware address by using system bus **101***d*, including from a ROM **101***c*, or module **101** can read module identity **110** from a nonvolatile memory such as a flash memory **101***w*. Step **514** could also take place after step **515** below. At Step **515**, module **101** can derive module private key **112** and a corresponding module public key **111** using (i) random number generator **128**, (ii) cryptographic parameters **126**, (iii) cryptographic algorithms **141**, and/or (iv) a key pair generation algorithm **141***e*. Module **101** at step **515** and elsewhere in the present invention can be a mobile phone such as, but not limited to, a smartphone. Private key **112** and corresponding module public key **111** can be derived according to a wide range of parameters **126**, and can utilize different algorithms for different pairs of keys, such as, but not limited to, RSA **153** or ECC **154**. Key derivation at step **515** could generate keys of various lengths, such as, but not limited to, 2048 bits with RSA **153** or 283 bits with ECC **154**, and other possibilities exist as well. If using ECC **154** to derive a pair of keys for module **101**, step **515** could also accommodate the use of different elliptic curves for compatibility with server **105**, such as, but not limited to, the use of odd-characteristic curves, Koblitz curves, and making sure the derived keys by module **101** use a compatible or identical elliptic curve or defined elliptic curve equation as server **105**, etc. Module **101** can use ECC parameters **137** or an ECC standard curve **138** in a parameters **126** to derive module private key **112** and/or module public key **111**.

**[0221]** Deriving keys in step **515** could also comprise using values such as constants or variables in a set of cryptographic parameters **126** to define an elliptic curve equation for use with an ECC algorithm **154**. The values or constants to define an equation for an elliptic curve could be input into a key pair generation algorithms **141***e* in the form of ECC parameters **137** or an ECC standard curve **138**. In an exemplary embodiment, where a parameters **126** does not include constants and variables for defining an elliptic curve equation, a key pair generation algorithms **141***e* could use pre-defined elliptic curves with ECC algorithms **154** such as, but not limited to, standardized, named curves in ECC standard curve **138** including exemplary values such as, but not limited to, sect283k1, sect283r1, sect409k1, sect409r1, etc. Exemplary, standardized named curves, as opposed to module **101** and server **105** using an internally generated elliptic curve equation using cryptographic parameters **126**, are also identified as example curves in IETF RFC 5480, titled "Elliptic Curve Cryptography Subject Public Key Information". Thus, module **101** could use either standardized elliptic curves, or a separate defined elliptic curve equation as specified in a parameters **126**.

**[0222]** The curve for module **101** to utilize in deriving module public key **111** and module private key **112** at step **515** could be specified in a set of cryptographic parameters **126**. Consequently, the parameters of keys generated by module **101** at step **515** (including key length or algorithms utilized) may be selected based upon the requirements of the application and can be included in a parameters **126**. When deriving keys at step **515**, module **101** may also preferably utilize data from sensor **101***f*, radio **101***z*, a bus **101***d*, a physical interface **101***a*, memory **101***e*, and/or a clock in order to generate a seed **129** for random number generator **128**, or random number generator **128** could utilize these inputs directly. A random number **128***a* can be input into key pair generation algorithm **141***e* in order to derive the module public key **111** and module private key **112**. Note that with ECC algorithms **154**, a module private key **112** can be a random number **128***a* in one embodiment, and the module public key **111** can be derived with a key pair generation algorithms **141***e* using the module private key **112** comprising the random number **128***a*.

**[0223]** Upon key derivation at step **515**, module private key **112** and module public key **111** can be recorded in a nonvolatile memory **101***w*. Module private key **112** can preferably not be transmitted or sent outside module **101**. Also note that over a potential lifetime of a decade or more of operation of module **101**, each time a new module private key **112** may be required (for various potential reasons outlined above), the external recording and/or transferring of module private key **112** incurs a potential security risk. Security risks can be compounded if the external location records private keys **112** for a plurality of modules **101**. Also, by internally generating private key **112** at step **515**, module **101** can overcome significant limitations and costs requiring the distribution of a pre-shared secret key Ki in the form of a SIM card or similar

physical distribution of a pre-shared secret key, after module **101** begins operations. In comparison, the use of a shared secret key **510** in the present invention does not require physical distribution of a new shared secret key **510** after module **101** begins operations. Module **101**'s key derivation could be triggered by either (i) a bootloader program **125**, where the bootloader program **125** determines that memory within module **101** does not contain a module private key **112**, or (ii) via a module instruction **502** such as, but not limited to, a "key generation" or "derive new keys" command in a response **209** from a server, and other possibilities exist as well.

**[0224]** Note that module **101**'s generation of keys after deployment and installation may create challenges for authentication of a new module public key **111** with module identity **110**, since module **101** may be connecting to server **105** or M2M service provider **108** via the Internet **107**. After module **101** creates new module public key **111** and module private key **112** at step **515**, at step **516** server **105** can receive a message **208** with the module identity **110**, the new module public key **111**, and cryptographic parameters **126**. Parameters **126** in message **208** at step **516** can represent the parameters **126** used to generate the module public key **111**. The sub-steps for a server **105** to receive a message **208** are also depicted and described in connection with FIG. **2** and also FIG. **1***c* above. Parameters **126** within a message **208** can comprise descriptive values for new module public key **111**. Note that at step **516**, server **105** does not need to receive new module public key **111** in the form of a certificate **122** (although it could be in the form of a certificate **122**). New module public key **111** could be received by server **105** within a string or field within a body **602** of a TCP/UDP packet **601***a*, illustrated in FIG. **6***b* below. As depicted in step **516** shown in FIG. **6***b* below, message **208** at step **516** can also optionally include a module public key identity **111***a*, which can be recorded in module database **105***k* along with module identity **110** and module public key **111***a*.

**[0225]** According to an exemplary embodiment, a first source (IP:port) number received in a first message **208** at step **516** can be different than a second source IP:port number in a second message **208** at step **518** below, wherein a response **209** send in step **519** below can preferably be sent to the second source IP:port number received in the second message **208** at step **518** in order to traverse a firewall **104** (as depicted and described in connection with packet **209***a* in FIG. **2**). In other words, the proper destination IP:port for a response **209** to a module **101** can change over time, such as the proper destination IP:port changing due to the use of sleep states by module **101** and/or function of a firewall **104**. Consequently, according to an exemplary embodiment, a response **209** can utilize a destination IP:port number equal to the source IP:port number received in the last (i.e. most recent) message **208** from module **101** received by server **105**.

**[0226]** At step **517**, server **105** can authenticate the message **208** received in step **516** using the shared secret key **510** described in step **513**. Server **105** could record the shared secret key **510** before step **517** in a module database **105***k*. If step **517** occurs for the first time in a lifetime of module **101**, then shared secret key **510** could comprise a pre-shared secret key **129***a* recorded by server **105** in a module database **105***k* illustrated in FIG. **1***f*. If step **517** occurs at subsequent time, then server **105** could have sent shared secret key **510** in a server encrypted data **504** and recorded shared secret key **510** in a module database **105***k* for later use (such as at step **517**). Server **105** can authenticate the message **208** according to message digest, or using the shared secret key **510** to process a symmetric key **127** within a symmetric ciphering algorithm **141***b*, where the successful encryption and decryption of data within message **208** using the shared secret key **510** on both ends could be confirmation that message **208** is authenticated, since both parties would only be able to mutually successfully encrypt and decrypt by sharing the same shared secret key **510**. As contemplated herein, the term "authenticating a public key" may refer to "authenticating a message that includes the public key", and both may refer to validating or verifying that a recorded module identity **110** stored in server **105** is associated with a receive module public key **111**.

**[0227]** Other possibilities exist as well for server **105** to use a shared secret key **510** in order to authenticate a message **208** that contains a new module public key **111** (where module **101** contains a new module private key **112**). In one embodiment, message **208** in step **516** could include a secure hash signature using secure hash algorithms **141***c*, where both the module **101** and the server **105** input a string combing at least a portion of the shared secret key **510** and a portion of the new module public key **111** into the secure hash algorithms **141***c* in order to obtain the secure hash signature. Module **101** could send the secure hash signature to server **105** in a message **208**. The authentication of a new module public key **111** in step **517** is also depicted and described in step **1202** of FIG. **12** below, including the authentication and/or verification of either (i) new module public key **111** or (ii) a message **208** that includes new module public key **111** according to steps that use alternatives to a shared secret key **510**. Thus, according to some exemplary embodiments (also discussed with step **1202** in FIG. **12** below), new module public key **111** can be authenticated and/or verified as being properly associated with a recorded module identity **110** in server **105** (*i*) without the use of a shared secret key **510**, and/or (ii) with alternatives to using shared secret key **510**. After receiving authenticated new module public key **111** in steps **516** and **517**, according to a preferred exemplary embodiment, server **105** can preferably only accept and process (A) either incoming (i) a symmetric keys **127** ciphered with a asymmetric ciphering algorithm **141***a*, and/or (ii) incoming server instructions **414**, when (B) the next or a subsequent incoming message **208** from module **101** using module identity **110** also includes a valid module digital signature **405** verified by using the new module public key **111**, received at step **516**.

**[0228]** According to an exemplary embodiment, shared secret key **510** can be associated with a module public key identity **111***a*, and shared secret key **510** can be used to authenticate a particular value for a module public key identity **111***a*. In this embodiment, (i) a message **208** with module public key **111** and a first module public key identity **111***a* may be authenticated using a shared secret key **510**, but (ii) a second message with module public key **111** and a second module public key identity **111***a* may not be authenticated using the same shared secret key **510**. Thus, in accordance with an exemplary embodiment, shared secret key **510** can be used for both (i) a single time for authenticating a module public key **111**, and (ii) authenticating a module public key **111** with a particular value for the module public key identity **111***a*. Note that module public key identity **111***a* can be particularly useful with key revocation, such that a key revocation could specify a particular module public key identity **111***a* (associated with a particular module public key **111**) to be revoked, but other module public keys **111** for a module

**101** and module identity **110** with different module public key identities **111***a* could remain valid and not revoked.

**[0229]** Although not illustrated in FIG. **5***b*, server **105** could operate with a certificate authority **118** in order to utilize a new module public key **111**, as described in this paragraph. In this case, server **105** could bypass the authentication at step **517**, but certificate authority **118** may perform step **517** in order to sign the certificate **122**, including possibly using shared secret key **510** to authenticate module public key **111**. At step **516**, new module public key **111** could be received by server **105** in the form of a uniform resource locator (URL) or domain name for download of a certificate **122** corresponding to the new module public key **111**. Using a certificate authority **118** in conjunction with step **516** is also depicted and described in connection with FIG. **5***b* of U.S. patent application Ser. No. 14/055,606, filed Oct. 16, 2013 in the name of John Nix, which is hereby incorporated by reference in its entirety.

**[0230]** After steps **516** and **517**, server **105** can update a module database **105***k* using the module identity **110** to insert or update the new module public key **111**, and parameters **126** associated with new module public key **111**. Server **105** may communicate with a plurality of modules **101**, and thus could utilize a module database **105***k* in order to record the new module public key **111** and parameters **126** with the module identity **110**. In one embodiment, the module identity **110** could preferably operate as an index within a table of module database **105***k* in order to speed reads and writes from the table used with module public key **111**, parameters **126**, and also selecting a symmetric key **127** for a symmetric ciphering algorithm **141***b* in later messages. As described in FIG. **1***g*, parameters **126** can include data useful for the operation of cryptographic algorithms **141** and module public key **111**. According to a preferred exemplary embodiment, some modules **101** in a system **100** could utilize a first elliptic curve, such as, but not limited to, using a first set of ECC parameters **137** or first ECC standard curve **138** within a parameters **126**, and other modules **101** could utilize a second and different elliptic curve within a parameters **126**, such as, but not limited to, a second set of ECC parameters **137** or second ECC standard curve **138**.

**[0231]** After verifying the new module public key **111** in a step **517**, at step **518** of FIG. **5***b*, server **105** could receive a second message **208**, and the second message **208** can include a module identity **110** and module encrypted data **403**. Although not illustrated in FIG. **5***b*, the second message **208** could also include a module digital signature **405**, wherein the module digital signature is created with the new module public key **111** received in step **516**. Server **105** could then utilize the steps illustrated in FIG. **4** in order to process the incoming message **208** with the new module public key **111**, including using the module identity **110** received in the second message **208** at step **518** to select the new module public key **111** and subsequently verify a module digital signature **405** using the new module public key **111** and digital signature algorithm **141***d*. Also as discussed in FIG. **4** in connection with processing a received message **208**, server **105** could decrypt the module encrypted data **403** in the second message **208** by using server private key **105***c*. In one embodiment, the second message **208** as illustrated in FIG. **5***b*, which could be the next message after authenticating module public key **111** in step **517**, could include a symmetric key **127**.

**[0232]** The module encrypted data **403** in step **518** could include a symmetric key **127** for utilization with a symmetric cipher **141***b*, where symmetric key **127** could be ciphered with an asymmetric ciphering algorithm **141***a*. In another embodiment, module **101** could also send sensor data in a module encrypted data **403** at step **518**. Or, at step **518** the second message **208** could be a signal and/or data (such as a random number **128***a*) for server **105** to use a key derivation function **141***f* with the server public key **114** and the new module public key **111** (received at step **516**) to create a new derived shared key **129***b* for use with symmetric ciphering algorithms **141***b* in subsequent messages **208**. In other words, in some embodiments derived shared key **129***b* can function as a symmetric key **127**. If the second message **208** in step **518** comprises a signal and/or data for server **105** to derive a new derived shared key **129***b*, then this second message **208** could then optionally leave off module encrypted data **403** and/or a module digital signature **405**. The successful use of a new derived shared key **129***b* (using the new module public key **111**, possible received in step **516**, and existing server public key **114**) with symmetric ciphering algorithms **141***b* at subsequent steps by both module **101** and server **105** can indicate to each the communications are mutually authenticated. Second message **208** could also include a server instruction **414**, a security token **401**, and/or a timestamp value **604***a*, and other possibilities exist as well without departing from the scope of the present invention.

**[0233]** At step **519**, server **105** can send a response **209** to module **101**, where the response **209** includes server encrypted data **504** and a module instruction **502**. Server **105** could take the steps to create and send response **209** as depicted and described in connection with FIG. **5***a*. Response **209** could be formatted according to the exemplary response **209** illustrated in FIG. **6***a* below. The module instruction **502** could be an acknowledgement **501** that the second message **208** sent in step **518** was received by server **105**. At step **520**, server **105** can receive a third message **208** with a confirmation **414** to server **105**. Confirmation **414** can be used to signal proper execution of module instruction **502** from a step **519**, if module instruction **502** comprised an instruction other than an "ACK" or acknowledgement **501**. In an embodiment where module instruction **502** in step **519** comprises an acknowledgement **501** from server **105**, then the confirmation **414** may omitted and in this case step **520** could be skipped.

**[0234]** At step **521** server **105** can determine or evaluate if a new module public key **111** and/or certificate **122** are required for continued operation. One reason for the need of new keys could be the expiration of a certificate **122** for module **101**, or the desire to utilize a different set of cryptographic parameters **126** such as, but not limited to, a longer key length for increase security or the use of a different ECC parameters **137** or a different ECC standard curve **138** with cryptographic algorithms **141**. As described elsewhere herein, many other possibilities exist for reasons why module **101** and/or server **105** can prefer for module **101** to utilize a new module public key **111** and new module private key **112**. Either server **105** or module **101** may determine that the use of a new module public key **111** and new module private key **112** may be preferred at step **521**. If module **101** determines that the use of a new module public key **111** and new module private key **112** is preferred or desirable, module **101** could send server **105***a* signal that new keys will be generated either before step **521** or at step **521**.

**[0235]** Upon determining new keys are desirable at step **521**, then server **105** could instruct module **101** to derive new private and public keys by returning to step **515**. Although not

illustrated in FIG. **5***b*, upon determining "yes" at step **521**, server **105** could send a module instruction **502** of "new key generation" and also a new or current set of cryptographic parameters **126** to utilize with the new module private key **112** and module public key **111**. In accordance with exemplary embodiments, module instruction **502**, including the "new key generation" instruction and set of parameters **126**, can be sent in a response **209** both (i) after module **101** wakes from a sleep or dormant state and sends a message **208** after waking from the sleep or dormant state, and (ii) before the expiration of a firewall port binding timeout value **117** after receiving the message **208**. If server **105** determines that new keys are not required or desirable at step **521**, server **105** can then proceed to step **312** and wait for additional incoming messages **208** from module **101** with module identity **110** or other modules. Step **312** is also depicted and described in connection with FIG. **3**. Other possibilities exist as well for a server to receive and respond to messages without departing from the scope of the present invention.

**[0236]** FIG. **6***a*

**[0237]** FIG. **6***a* is a simplified message flow diagram illustrating an exemplary message received by a server, and an exemplary response sent from the server, in accordance with exemplary embodiments. FIG. **6***a* illustrates exemplary details within message **208** received by server **105** and also response **209** sent by server **105**. Message **208** may comprise a TCP/UDP packet **601***a* sent from module **101** source IP:port **204** to server **105** destination IP:port **207**. According to an exemplary embodiment, UDP or UDP Lite formatting for TCP/UDP packet **601***a* may be preferred. Source IP:port **204** and destination IP:port **207** in message **208** may be included within a header in TCP/UDP packet **601***a*. Although a single message **208**, response **209**, module **101**, and server **105** are shown in FIG. **6***a*, system **100** as illustrated in FIG. **2** and other systems depicted herein may comprise a plurality of each of the nodes and datagrams illustrated in FIG. **6***s*. As contemplated herein, the term "datagram" may also refer to a "packet", such that referring to as datagram **601***a* can be equivalent to referring to packet **601***a*. Note that when using TCP protocol, a packet within a series of TCP messages can also be a datagram **601***a*.

**[0238]** TCP/UDP packet **601***a* may include a body **602**, which can represent the data payload of TCP/UDP packet **601***a*. The data payload of message **208** can optionally include channel coding **406** as described in FIG. **4** above, if the transport protocol for TCP/UDP packet **601***a* supports the transmission of bit errors in the body **602** (as opposed to entirely dropping the packet), such as, but not limited to, with the UDP Lite protocol. Support for the transmission of bit errors in body **602** by wireless network **102** would be preferred over entirely discarding a packet, since the programs such as module controller **105***x* could include support for and utilization of channel coding **406**. Without UDP Lite formatting, message **208** can alternatively sent by module **101** as a UDP datagram, such as if wireless network **102** (or a wired connection) does not support the UDP Lite protocol.

**[0239]** Note that if (A) message **208** comprises (i) regular UDP or TCP formatting (i.e. not UDP Lite or similar variations) within an IPv6 network, or (ii) a UDP or TCP format within an IPv4 network with a checksum **603** enabled (i.e. checksum **603** not equal to zero), then (B) channel coding **406** may optionally be omitted. Checksum **603** can comprise a value to for an integrity check of a packet **601***a*, and the calculation and use of checksum **603** is defined in IETF standards for TCP and UDP packets. In accordance with an exemplary embodiment, including the use of IPv6 for Internet **107** and a UDP datagram for message **208** and response **209**, a checksum **603** sent by module **101** in a message **208** does not equal a checksum **603** in the message **208** received by server **105**.

**[0240]** The body **602** can include a module identity **110**, module encrypted data **403**, and channel coding **406**. Although not illustrated in FIG. **6***a*, body **602** could also include a module digital signature **405**, as illustrated in FIG. 6 of U.S. patent application Ser. No. 14/039,401. Module identity **110** is illustrated in FIG. **6***a* as external to module encrypted data **403**, although module identity **110** may optionally only be included in module encrypted data **403**, and in this case module identity **110** would not be external to module encrypted data **403** in a body **602**. By including module identity **110** as external to module encrypted data **403**, server **105** can use the unencrypted module identity **110** in order to select either (i) the appropriate module public key **111** to verify module digital signature **405** if an asymmetric cipher **141***a* is used within cryptographic algorithms **141**, or (ii) the appropriate symmetric key **127** within cryptographic algorithms **141** to decrypt the module encrypted data **403**. Module public key **111** and symmetric key **127** may preferably be recorded in a module database **105***k*, such that server **105** can access a plurality of public keys associated with different module identities **110** with different bodies **602** for a plurality of modules **101**.

**[0241]** Thus, by including module identity **110** external to module encrypted data **403**, server **105** can utilize the module identity **110** to query a module database **105***k* and select the appropriate module public key **111** or symmetric key **127**. As noted previously, module identity **110** could comprise a string or number that is uniquely associated with module identity **110**, such as, but not limited to, a session identity, as opposed to being a module identity **110** that is read from hardware in module **101** such as, but not limited to, an IMEI number, Ethernet MAC address, etc. Module identity **110** is illustrated in FIG. **6***a* as a session identity that is a different representation of module identity **110** of a serial number such as in FIG. **2**, but in both cases the values can comprise a module identity **110** since the values can be uniquely associated with module **101** at different points in time.

**[0242]** According to an exemplary embodiment where asymmetric ciphering **141***a* of module encrypted data **403** is utilized, such as (i) the first message **208** sent by module **101** and (ii) where a symmetric key **127** had not been previously exchanged, module identity **110** can be (a) within module encrypted data and (b) not external to module encrypted data **403**. In this case, server **105** can utilize server private key **105***c* to, in sequence, decrypt module encrypted data **403**, extract module identity **110** from the decrypted module encrypted data **403**, and then used the module identity **110** to select module public key **111** from module database **105***k* in order to verify a module digital signature **405**. In a related embodiment, if a module identity **110** is in body **602** and external to module encrypted data **403**, then module identity **110** could be obfuscated or otherwise ciphered according to a pre-agreed algorithm with server **105**, such that server **105** can utilize the obfuscated or ciphered module identity **110** to select a module public key **111** from module database **105***k*. The value of "[Module Identity String]" shown in FIG. **6***a* could comprise an obfuscated module identity **110**. According to an exemplary embodiment where (i) symmetric cipher-

ing **141***b* of module encrypted data **403** is utilized, such as after a first message **208** had already been sent by module **101** and a symmetric key **127** had previously been exchanged, then (ii) module identity **110** can be external to module encrypted data **403** and in body **602** in order for server **105** to utilize module identity **110** and select symmetric key **127** from a module database **105***k*, thereby enabling server **105** to decrypt the module encrypted data **403** using the selected symmetric key **127** and a symmetric ciphering algorithm **141***b*.

**[0243]** In exemplary embodiments, a module digital signature **405** may optionally be omitted from body **602** after module **101** has previously sent symmetric key **127** in a previous message **208** to the message **208** illustrated in FIG. **6***a*. In other words, in a series of messages **208**, module **101** can preferably change from (i) using asymmetric ciphering **141***a* with in a previous message **208** that includes symmetric key **127** in a module encrypted data **403** (where the initial message **208** also includes module digital signature **405** and module identity **110**) to (ii) using symmetric ciphering **141***b* with subsequent messages **208** without module digital signature **405** in the series (where the subsequent messages **208** can optionally include an obfuscated module identity **110** external to module encrypted data **403** for server **105** to select the appropriate symmetric key **127**). Message **208** illustrated in FIG. **6***a* can comprise a subsequent message **208** as described in the previous sentence. A series of messages **208** could begin when the initial message **208** is sent by module **101** and end when expiration time **133** of symmetric key **127** has transpired, and subsequently a new series of messages **208** could begin where the first message **208** in the new series of messages changes back to asymmetric ciphering **141***a* with initial message **208** that includes symmetric key **127** in a module encrypted data **403** (where the initial message **208** also includes a new module digital signature **405**). An example of the initial message **208** described in this paragraph can comprise message **208** illustrated in FIG. 6 of U.S. patent application Ser. No. 14/039,401, filed Sep. 27, 2013 in the name of John Nix, which is hereby incorporated by reference in its entirety. Other possibilities exist as well without departing from the scope of the present invention.

**[0244]** Using a message **208** with a module digital signature **405** can be both more efficient and overall more secure than digest authentication (such as the digest authentication described in IETF RFC 2069), although using digest-based authentication may be alternatively used. The use of a module digital signature **405** requires only a single packet for message **208** and a single packet for response **209** for secure communication between module **101** and server **105**. Module encrypted data **403** illustrated in FIG. **6***a* can be processed using the steps and algorithms described in FIG. **4**. Note that module encrypted data **403** as illustrated in FIG. **6***a* is shown in a plaintext form for ease of illustration, but actual module encrypted data **403** within body **602** of a packet **601***a* could be transmitted as binary, hexadecimal, Base64 binary-to-text encoding, or other encoding rules and strings of the actual data within module encrypted data **403** would not normally be human readable.

**[0245]** In an exemplary embodiment, encryption by module **101** may optionally be omitted, and the server instruction **414** with corresponding data could be included within a message **208** without encryption, such as if security could be maintained at the network level. As one example for this embodiment without encryption, server instruction **414** could be included in body **602** as plaintext. The encryption and/or security could be applied through other means, such as, but not limited to, a secure tunnel between module **101** and server **105**, although setting up and maintaining a secure tunnel and similar or other means of security may require more processing and bandwidth resources than the efficient techniques described herein.

**[0246]** Module encrypted data **403** can include a server instruction **414**, a server identity **206**, a module identity **110**, a security token **401**, a timestamp **604***a*, and a sensor measurement **604***b*. The server instruction **414** can represent the purpose of the message **208** for server **105**, and FIG. **6***a* illustrates an "update" for server instruction **414**. An update for server instruction **414** could be used to periodically notify server **105** of regular, periodic sensor measurements **604***b* acquired by a sensor **101***f* or also data from a plurality of sensors. An update for server instruction **414** may also comprise a periodic report regarding monitored unit **119**, and a server instruction **414** is described in FIG. **4**. Other server instructions **414** besides an "update" may be included in a module encrypted data **403** within a body **602**. The "update" illustrated in message **208** in FIG. **6***a* can also include a new symmetric key **127**, and the module encrypted data **403** illustrated in FIG. **6***a* may comprise the use of either an asymmetric ciphering **141***a* with public/private keys, or (ii) symmetric ciphering **141***b* with a symmetric key **127**.

**[0247]** An initial transmission or negotiation of a symmetric key **127** may preferably utilize asymmetric ciphering **141***a* and the use of a public key as an encryption key and a private key as a decryption key. Subsequent transmission of a new symmetric key **127** may utilize either (i) a symmetric cipher **141***b* with a previously negotiated but still valid symmetric key **127** (i.e. expiration time **133** has not transpired), or (ii) asymmetric ciphering **141***a*. If the data within instruction **414** is longer than the maximum data length supported by a selected asymmetric ciphering algorithm **141***a* and the public/private key pair, then module encrypted data **403** within message **208** can be broken up into several sections, such that the data within each section is less than the maximum data length supported by the asymmetric ciphering algorithm **141***a* and key length. In an exemplary embodiment, a first symmetric key **127** can be used with module encrypted data **403** and a second symmetric key **127** can be used with server encrypted data **504**. The first symmetric key **127** and second symmetric key **127** can be different, including using a first symmetric ciphering algorithm **141***b* with the first symmetric key and a second symmetric ciphering algorithm **141***b* with the second symmetric key **127**. In another exemplary embodiment, in order to reduce the number of messages required to be transmitted and thus save power usage by a module **101**, symmetric key **127** used with module encrypted data **403** and server encrypted data **504** can be the same and rotated periodically such, but not limited to, when expiration time **133** for a symmetric key **127** transpires.

**[0248]** Module identity **110** within module encrypted data **403** can represent the identity of module **110**, and could represent a serial number read by module **101** from a read-only hardware address. Module identity **110** is described in FIG. **1** *c* and can represent a unique identifier of module **101**. Module identity **110** outside module encrypted data **403** can represent a string or number that is different than a serial number that can be used by module **101** within a module encrypted data **403**. Security token **401** within module encrypted data **403** can represent a random string in order to

make message **208** reasonably unique and thus system **100** in FIG. **2** and other systems illustrated herein robust against replay attacks. Security token **401** is described in FIG. **5***a*. Timestamp **604***a* can represent a time value that module **101** sends message **208** or a time value that module **101** acquired sensor data **604***b*. Sensor data **604***b* is described with the description of a sensor **101***f* in FIG. **1***e*, and sensor data **604***b* can represents data module **101** acquires using sensor **101***f*. Sensor data **604***b* within message **208** may be stored by server **105** in a module database **105***k*, or potentially forwarded to another server such as, but not limited to, an application server **171** for additional processing. Sensor data **604***b* can comprise a wide range of values for a sensor **101***f* besides the exemplary value of a temperature reading shown in FIG. **6***a*, including raw sensor data, compressed sensor data, and processed or averaged data. The specific sensor data **604***b* shown in FIG. **6***a* is illustrated to be exemplary and not limiting for sending and receiving sensor data. Sensor data **604***b* may also be referred to as a sensor measurement **604***b*.

**[0249]** FIG. **6***a* also illustrates exemplary details within response **209** sent by server **105**. Response **209** may comprise a TCP/UDP packet **601***b* sent from server **105** IP:port **207** the IP address **210** and port number **605**, where IP address **210** represents the external IP address of wireless network firewall **104** and port number **605** is the source port in message **208** as received by server **105** (i.e. the source port in message **208** after traversing the firewall **104** illustrated in FIG. **6***a*). Thus, IP:port with IP address **210** and port number **605** in response **209** may be different than IP:port **204** in message **208**, since the presence of a wireless network firewall **104** may perform NAT routing, which could change the source IP address and source port number from IP:port **204** to IP address **210** and port number **605** in message **208**, as received by server **105**. The use of wireless network firewall **104** in wireless network **102** may require that response **209** be sent from IP:port **207** to IP address **210** and port number **605** in order to be properly processed by firewall **104** and forwarded to module **101** at IP:port **204**. Source IP:port **207** and destination IP address **210** and port number **605** in response **209** may be included within a header in TCP/UDP packet **601***b*, as illustrated in FIG. **6***a*. TCP/UDP packet **601***b* could comprise a regular UDP packet, a UDP Lite packet, or a TCP datagram, or similar protocols supported by an Internet **107**. TCP/UDP packets **601***a* and **601***b* may utilize the same protocol.

**[0250]** As noted previously, the use of checksums may be mandatory in IPv6 networks, and thus a response **209** comprising a packet **601***b* can include a checksum value **603** (illustrated in message **208** but not response **209**) for the header. The use of firewalls such as firewall **104** can change the header values in a packet **601***b*. In accordance with a preferred exemplary embodiment, a first checksum value **603** within a response **209** sent by server **105** can be different and/or not equal to a second checksum value **603** within the response **209** received by module **101**. Likewise, in an exemplary embodiment, a first checksum value **603** within a message **208** sent by a module **101** can be different and/or not equal to a second checksum value **603** within the message **208** received by server **105**.

**[0251]** A UDP, TCP, or UDP Lite datagram as a TCP/UDP packet **601***b* within response **209** may include a body **606**. Body **606** may comprise the payload or data within a UDP, TCP, or UDP Lite packet. Body **606** can include a server identity **206**, a server digital signature **506** (not shown in FIG. **6***a*), server encrypted data **504**, and channel coding **406**. Server identity **206** is illustrated in FIG. **6***a* as external to server encrypted data **504** within body **606**, but server identity **206** may optionally be included in server encrypted data **504** instead. Module **101** may communicate with a plurality of servers **105**, and server identity **206** as external to server encrypted data **504** can allow module **101** to select the appropriate symmetric key **127** to utilize for decrypting server encrypted data **504** (since each of the multiple servers **105** that module **101** communicates with may utilize a different symmetric key **127**).

**[0252]** Also note that the server identity **206** can be similar to module identity **110**, such that multiple different values for server identity **206** could be utilized in different systems illustrated herein, but each of the different values could preferably be uniquely associated with a server **105**. As one example, server identity **206**, outside server encrypted data **504** as illustrated in FIG. **6***a*, may comprise a session identity or session identifier, as opposed to a different server identity **206** that could comprise a hardware serial number or domain name for server **105**. Thus, server identity **206** outside a server encrypted data **504** may be a different string or representation than server identity **206** within server encrypted data **504**, but both strings/numbers used for server identity **206** in response **209** could be associated with server **105**. In an exemplary embodiment, a set of servers **105***n* can collectively use a server identity **206**.

**[0253]** Although not illustrated in FIG. **6***a*, a server digital signature **506** in body **606** can comprise a secure hash signature of a subset of body **606**, where the subset of body **606** can comprise server encrypted data **504**, and/or server identity **206** as illustrated in FIG. **6***a*. The use of a server digital signature **506** in a body **606** is illustrated in FIG. 6 of U.S. patent application Ser. No. 14/039,401, filed Sep. 27, 2013 in the name of John Nix, which is hereby incorporated by reference in its entirety. In this manner, module **101** can utilize server digital signature **506** to authenticate that response **209** was sent by server **105**. Channel coding **406** in body **606** is also depicted and described in connection with FIG. **5***a* above. The server digital signature **506** may optionally be omitted as well.

**[0254]** Body **606** may include server encrypted data **504**. Server encrypted data **504** is depicted and described in connection with FIG. **5***a* above. Server encrypted data **504** may include an acknowledgement **501**, wherein acknowledgement **501** can notify module **101** that message **208** has been received by server **105**. As illustrated in FIG. **6***a*, server encrypted data **504** may optionally also include a module instruction **502** for module **101**. The module instruction **502** could be a string that contains instructions or configuration parameters for module **101**, such as an order to change state, parameters regarding the monitoring of monitored unit **119**, server names or addresses, radio frequency parameters, timer values, settings for actuator **101***y*, etc. A module instruction **502** is depicted and described in connection with FIG. **5***a* above. The exemplary module instruction **502** illustrated in FIG. **6***a* comprises a "key generation" **608** instruction for module **101** derive a new set of keys, also depicted and described in connection with FIG. **5***b* above. Other possibilities for a module instruction **502** within a response **209** are possible as well without departing from the scope of the present invention. Although not depicted in FIG. **6***a* or FIG. **2**, if response **209** includes a module instruction **502**, according to an exemplary embodiment, module **101** can preferably send a second message **208** to server **105**, where the second

message **208** includes a confirmation that module instruction **502** was successfully executed or implemented by module **101**. This confirmation could be included in a server instruction **414** for server **105** within a second message **208**, and the confirmation could include a timestamp value **601***a* for when the module instruction **502** was executed. A timestamp value **601***a* may be useful for tracking time of actions and data collected, when a module **101** may only periodically have access to a network **102** and also may periodically be dormant or sleep.

**[0255]** Also, although a server encrypted data **504** may be included within a body **606** in exemplary embodiments, body **606** may optionally omit server encrypted data **504** and include data from server **105** or a set of servers **105***n* that is not encrypted, such as, but not limited to, plaintext. As one example in this case, acknowledgement **501** could be included in body **606** as plaintext. Also, although not illustrated in FIG. **6***a*, server encrypted data **504** could include a symmetric key **127** for module **101** to utilize with symmetric ciphering **141***b* in cryptographic algorithms **141** for processing a module encrypted data **403** in subsequent messages **208** and/or responses **209**. Server encrypted data **504** in a response **209** may include a security token **401**. Security token **401** may be a random string and may also be generated by either server **105** or module **101**. If security token **401** is generated by module **101**, then security token **401** may be included in message **208** and also utilized by server **105** in response **209**, as illustrated in FIG. **6***a*. Other possibilities exist as well without departing from the scope of the present invention.

**[0256]** FIG. **6***b*

**[0257]** FIG. **6***b* is a simplified message flow diagram illustrating an exemplary message received by a server, wherein the message includes a derived module public key, in accordance with exemplary embodiments. As discussed in FIG. **5***b*, there can be cases where module **101** derives a new module public key **111** and new module private key **112**. On example would be the initial creation of the key pairs by module **101**, and many other examples could exist as well. FIG. **6***b* can illustrate an exemplary format and contents of a message **208** for steps **516** and **517** of FIG. **5***b*. This exemplary message **208** can also help to illustrate the significant differences from conventional technology and improvements for efficient and secure communications by utilizing embodiments contemplated herein.

**[0258]** A message **208** illustrated in FIG. **6***b* using steps **516** and **517** can include (i) sending new module public key **111**, a module public key identity **111***a*, a module identity **110**, a server instruction **414**, a security token **401**, and a set of cryptographic parameters **126** associated with the new module public key **111** and/or cryptographic algorithms **141** for using the new module public key **111**. Exemplary cryptographic parameters **126** illustrated in FIG. **6***b* include (i) a secure hash algorithm **141***c* to utilize in signatures, which could comprise the SHA 256 algorithm as shown (which may also be known as the SHA-2 algorithm), (ii) a selected elliptic curve for use with ECC algorithms **154** or a modulus to use with RSA algorithms **153**, and (iii) a time-to-live value for the public key, such as, but not limited to, the illustrated "time to live" value of 1 year shown in FIG. **6***b*. The time value for the validity of new module public key **111** could alternatively be specified in a set expiration date. Other values associated with cryptographic algorithms **141** could be included in a set of cryptographic parameters **126** as well, and the illustrated values are intended to be exemplary instead of limiting. In exemplary embodiments, the set of cryptographic parameters **126** in a message **208** could comprise a set of cryptographic parameters **126** depicted and described in connection with step **1105** of FIG. **11** below, and/or FIG. **1***g*.

**[0259]** Additional values or fields within a message **208** associated with communicating a new module public key **111** with server **105** could include a server instruction **414** of "new public key". This server instruction **414** could inform server **105** to utilize the new module public key **111** within the message **208**. Module public key identity **111***a* can include a sequence number or identity for the new module public key **111**, such that module **101** or server **105** can properly reference and/or record the key from a plurality of module public keys **111** that could be associated with module identity **110**. Although module public key identity **111***a* is illustrated as a separate field in server instruction **414**, module public key sequence number **111***a* could optionally be included in a set of cryptographic parameters **126**, such that the value within cryptographic parameters **126** specifies the current sequence number of module public key identity **111***a* for the new module public key **111** included in a message **208**.

**[0260]** Other fields and features within a message **208** as illustrated in a FIG. **6***b* can be similar to the fields presented in FIGS. **6***a*. Since (a) FIG. **6***b* can also illustrate a first message **208** sent by a module **101** to a server **105**, such as after keys are derived in a step **515**, then (b) module **101** can read multiple values from RAM **101***e* or a nonvolatile memory **101***w* or **101***c* in order properly construct or format message **208**. Each of (i) destination IP:port number **207**, (ii) parameters **126**, and (iii) shared secret key **510** can preferably be written into nonvolatile memory at step **512** of FIG. **5***b*, if message **208** in FIG. **6***b* represents the first message **208** sent by module **101**. The source IP:port number **204** can represent a number assigned by an operating system **101***h*.

**[0261]** If message **208** in FIG. **6***b* comprises a subsequent time message **208** is received by server **105** (i.e. not a first time module **101** sends a module public key **111**), such as after step **521** illustrated in FIG. **5***b*, then each of (i) destination IP:port number **207**, (ii) parameters **126**, and (iii) shared secret key **510** could be updated by server **105** using a module instruction **502** within a server encrypted data **504** before message **208** illustrated in FIG. **6***b* is received by server **105**. In this manner, shared secret key **510** could change from (i) comprising a pre-shared secret key **129***a* (for a first message **208** after initial module key derivation) to (ii) comprising a shared secret key that is sent by server **105** within a server encrypted data **504** (for a subsequent message **208** after a subsequent module key derivation).

**[0262]** After receiving message **208**, server **105** can use the module identity **110** illustrated in a body **602** of FIG. **6***b* to select the shared secret key **510** in order authenticate message **208**. As described in step **517** of FIG. **5***b*, server **105** may preferably authenticate message **208** that includes module public key **111** in order to confirm that module public key **111** originated from physical module **101** with a hardware module identity **110** (as opposed to being an imposter submitting the module public key **111**). The use of a channel coding **406** is described in connection with FIGS. **4** and **5***a*, and channel coding may optionally be omitted. If message **208** comprises a UDP Lite packet, then channel coding may optionally be applied within the body **602**. If message **208** comprises a UDP packet, then channel coding may comprise sending the exact same UDP packet **601***a* multiple times, such as, but not limited to, an exemplary 3 packets **601***a* sent at the same time.

**[0263]** Although not illustrated in FIG. **6***b*, in an exemplary embodiment module public key **111** could also be received in a message **208**, where the module public key **111** and parameters **126** can be included in an encrypted format within a module encrypted data **403**. As depicted and described in connection with steps **1001** and **1002** of FIG. **10**, and also FIG. 11 of U.S. patent application Ser. No. 14/039,401, the security of a system **100** and other systems illustrated herein can be further increased by both (i) ciphering module public key **111** and the set of cryptographic parameters **126**, and (ii) only sharing the module public key **111** in a confidential manner with server **105** and/or a set of servers **105***n*. If module **101** needed a module public key **111** for other purposes, such as, but not limited to, obtaining a certificate, then a second, publicly disclosed module public key **111** could be utilized, where the second module public key **111** is different than a module public key **111** using parameters **126** that is sent to a server **105** in a module encrypted data **403**.

**[0264]** FIG. **6***b* also illustrates an exemplary embodiment, where module public key **111** can be authenticated with server **105** using a module digital signature **405**. If message **208** comprises a first time module **101** utilizes a step **516** and step **517**, such that a module public key **111** has not previously been sent to server **105** and/or a set of servers **105**, then in an exemplary embodiment message **208** could include a module digital signature **405** using the shared secret key **510**, which could comprise the pre-shared secret key **129***a*. If message **208** comprises a subsequent time module **101** utilizes a step **516** and step **517**, such that a module public key **111** has previously been sent to server **105**, then message **208** could include a module digital signature **405** using the previous module private key **112** (i.e. not the new module private key **112** associated with the new module public key **111** in the message **208** shown in FIG. **6***b*). As noted in FIG. **5***b*, module digital signature **405** could be omitted, and message **208** with module public key **111** could be authenticated using a message digest algorithm and the shared secret key **129***a*. Other possibilities for a module **101** to send a new module public key **111** in a message exist as well without departing from the scope of the present invention.

**[0265]** FIG. **7**

**[0266]** FIG. **7** is a simplified message flow diagram illustrating an exemplary system with exemplary data transferred between a module and an application using a server, in accordance with exemplary embodiments. FIG. **7** includes a system **700** and illustrates an exemplary message **208** from a module **101** to a server **105** and also an exemplary application message **701** between an application **171***i* and server **105**. Note that application message **701** could also be considered as transferred between, sent to, or received from server **105** and application server **171**. System **700** can comprise a module **101**, a server **105**, and an application **171***i* operating on an application server **171**, and these elements may communicate over a network such as, but not limited to, the Internet **107**. For example, application server **171** may utilize an IP:port number **702** for sending and receiving messages with server **105**. The IP address within IP:port number **702** is illustrated as an IPv4 address, but an IPv6 address could be utilized as well, or other addressing schemes are possible. Message flows within a module **101** from a sensor **101***f* and to an actuator **101***y* are also included in a system **700** as illustrated in FIG. **7**. Message flows within a module **101** could utilize a system bus **101***d*.

**[0267]** As illustrated in FIG. **7**, before module **101** sends a module public key **111** to server **105**, possibly by using step **516** as illustrated in FIG. **7**, module **101** can derive the public and private keys using step **515** and a set of cryptographic parameters **126**. Alternatively, in a different embodiment module **101** may have the module public key **111** and module private key **112** generated outside module **101** and loaded into a non-volatile memory **101***w*, and in this case step **515** shown can be optionally omitted. Server **105** can utilize step **516** to receive a module public key **111** from module **101**. Server **105** can utilize a step **517** and a shared secret key **510** to authenticate a message **208** that contains the module public key **111** from step **516**. Authentication of module public key **111** may be preferred in order to ensure that the module public key **111** is properly associated with the correct physical module **101**, and prevent an imposter, hacker, etc. from sending in a fake module public key **111** for module **101**. After using step **517** to authenticate module public key **111**, server **105** can record module public key **111** and associated module identity **110** (plus optionally a module public key identity **111***a*) in a module database **105***k*. Although not illustrated in FIG. **7**, server **105** can also send an application message **701** to application **171***i* after successfully recording module public key **111**.

**[0268]** Application **171***i* operating within an application server **171** can send an application message **701** to server **105**, and server **105** can receive the application message **701**. Application message **701** could include a module instruction **502**, where the module instruction **502** could comprise an actuator setting **706**. Although not illustrated in FIG. **7**, module instruction **502** as transmitted or sent by application **171***i* or application server **171** could include a module identity **110** and/or an actuator identity **152**. Actuator setting **706** could include a setting value and an actuator identity **152**. As discussed below in connection with FIG. **8**, actuator setting **706** within an application message **701** could be received within a secure connection data transfer **802** from application server **171**. Thus, in an exemplary embodiment, the actuator setting **706** may preferably not be plaintext as transmitted across a network such as, but not limited to, the Internet **107** between server **105** and application server **171** in an application message **701**.

**[0269]** A module instruction **502** (*i*) from an application **171***i* or application server **171**, and (ii) within an application message **701** could include other exemplary values or instructions for a module **101**, besides the exemplary actuator setting **706**. According to exemplary embodiments, a module instruction **502** could comprise information for module **101** such as (i) sleep timers or instructions or values for a CPU wake controller **101***u*, (ii) server address **106** or server identity **206** for communicating with a server **105** (such as sending a different server address **106** for module **101** to utilize in future communications), (iii) a new or updated values for set of data reporting steps **101***x*, (iv) a new or updated module program **101***i*, (v) software or firmware for operating system **101***h* and device driver **101***g* (including a pointer or reference to a location where the updated module program **101***i* could be located), (vi) a calibration value for sensor **101***f* or actuator **101***y*, (vii) values for a set of cryptographic parameters **126**, (viii) software or settings for radio **101***z*, (ix) updated cryptographic algorithms **141**, (x) a new module private key **112**, (xi) a symmetric key **127**, (xii) a pre-shared secret key value **129***a* for use in communicating with a wireless network **102** (where the pre-shared secret key value **129***a* can be the

equivalent of a Ki value in a network supporting ETSI/3GPP standards), (xii) a value for a module identity **110**, (xiii) a value to use in a channel coding **406**, (xiv) a security token **401** or settings for using security tokens, and/or (xv) values for a electronic UICC (eUICC). Other possibilities exist as well for a module instruction **502** without departing from the scope of the present invention. After receiving module instruction **502** in a response **209** from server **105**, module **101** could record the data in module instruction **502** within a nonvolatile memory **101***w* or RAM **101***e*. In an exemplary embodiment, a eUICC received within a module instruction **502** by module **101** could provide the data and parameters for module **101** to connect with another wireless network **102**, which could comprise a second PLMN.

**[0270]** After receiving application message **701**, server **105** can wait for wait interval **703**. As depicted and described in connection with FIGS. **2** and **6***a*, firewall **104** may be present in a system **700** and/or other systems depicted in the present invention, and a firewall **104** could block the transmission or sending of packets from server **105** to module **101** at arbitrary times. In addition, according to exemplary preferred embodiments, module **101** can enter periods of sleep or dormancy using a CPU wake controller **101***u* in order to conserve energy or the life of a battery **101***k*, if present. During periods of sleep or dormancy, module **101** may not be able to receive packets from server **105**. Consequently, server **105** can preferably wait for the wait interval **703** as illustrated in FIG. **7**, before sending response **209** which could include the module instruction **502**. As illustrated in FIG. **7**, the module instruction **502** could include an actuator setting **706**, but module instruction **502** could include other data as well such as the exemplary module instructions **502** described in the previous paragraph.

**[0271]** According to exemplary embodiments, wait interval **703** can vary depending upon module **101** and monitored unit **119**, and wait interval **703** could comprise a wide range of values. Module **101** could send a sensor data **604***b* or a report or a message **208** at exemplary reporting intervals such as, but not limited to, every minute, 10 minutes, hour, 6 hours, daily, or longer. Wait interval **703** could be associated with the reporting interval, and the wait interval **703** would end when the next message **208** from module **101** is received. If server **105** supports a plurality of modules **101**, wait interval **703** can be associated with the specific module **101** associated with the module instruction **502**, possibly by using a module identity **110** in both a message **208** and an application message **701**. In other words, server **105** can preferably wait for a message **208** from the specific module **101** associated with the module instruction **502** before sending the response **209** which could include the module instruction **502**. Response **209** could be sent using the source and destination IP:port numbers depicted and described in connection with FIG. **2**.

**[0272]** Upon the receipt of message **208** from module **101** with module identity **110**, the wait interval **703** can end. As illustrated in FIG. **7**, message **208** could include a server instruction **414**. The server instruction **414** in the exemplary embodiment illustrated in FIG. **7** comprises an "update" server instruction **414**, and could include a sensor measurement **604***b* and/or a timestamp **604***a*. Sensor measurement **604***b* could be obtained by module **101** from sensor **101***f* before sending message **208**, and possibly after module **101** wakes from a dormant state using a CPU wake controller **101***u*. Sensor measurement **604***b* could be collected by a module program **101***i* using a system bus **101***d*. As illustrated in FIG. **6***a*, a server instruction **414** with sensor data **604***b* could be within a module encrypted data **403** and received by server **105**. Server **105** could utilize the steps illustrated in FIG. **4** to process the received message **208** at the end of wait interval **703**. Sensor measurement **604***b* as used by module program **101***i*, server **105**, application **171***i*, and/or application server **171** could represent a different string or number at each element, depending upon encoding rules or encoding schemes utilized by each element, but sensor measurement **604***b* at each location can represent data or a value collected by a sensor **101***f*.

**[0273]** After processing the received message **208** that could include sensor data **604***b* and/or timestamp **604***a*, server **105** can send application **171***i* operating on application server **171** an application message **701** that includes an update instruction **704**, where update instruction **704** could include sensor data **604***b*, module identity **110**, and sensor identity **151**, if present. Update instruction **704** could include data other than sensor data **604**, such as data pertaining to the state of module **101**, including subcomponents illustrated in FIGS. **1***b* and **1***e*. Using update instruction **704** or a plurality of update instructions **704**, application **171***i* can aggregate data to generate reports for presentation to user **183** or make decisions using service controller **171***x*. Based on data input in multiple update instructions **704** over time, application **171***i* could output module instruction **502** in an application message **701**. Application **171***i* could record data received in update instruction **704** within an application database **171***k*, and process the data using a service controller **171***x* in order to automatically generate module instructions **502** using a plurality of sensor data **604***b* for a module **101** or a set of modules **101**.

**[0274]** After receiving message **208** with server instruction **414**, server **105** can send a response **209** to module **101**. Note that response **209** is illustrated in FIG. **7** as being sent after sending update instruction **704** to application server **171**, but response **209** could also be sent to module **101** before sending update instruction **704** to application server **171**. Response **209** can include module instruction **502**, where module instruction **502** could comprise the actuator setting **706** server **105** received from application **171***i*, according to an exemplary embodiment. Module instruction **502** could also comprise other data besides actuator setting **706** for module **101** in other exemplary embodiments, as outlined above. Although not illustrated in FIG. **7**, response **209** could include module instruction **502** within a server encrypted data **503** using the steps depicted and described in connection with FIG. **5***a*. Module instruction **502** could also include actuator identity **152** associated with actuator setting **706**. Response **209** can be formatted as depicted and described in FIGS. **2** and **6***a*, such that response **209** can traverse a firewall **104** and be received by module **101** using IP address **204**. Network firewall **104** is illustrated as a dashed line in FIG. **7**, and may be optionally not be present. But, the use of network firewall **104** may be included in a system **100** and/or system **700** and the presence and operation of a network firewall **104** may be beyond the control of a module **101**, server **105**, module provider **109**, M2M service provider **108**, etc., and thus a system **700** can support a firewall **104** in exemplary embodiments.

**[0275]** After receiving response **209** with the module instruction **502** and actuator setting **706**, module **101** can process the response **209**, which could also include server encrypted data **503**. Module **101** could extract or read actua-

tor setting **706** from the module instruction **502**. Module instruction **502** could include an actuator identity **152**. Module **101** can use a module program **101***i* to send the actuator setting **706** to the actuator **101***y* with actuator identity **152**. Actuator setting **706** as sent by module program **101***i* may be in a different format or data structure than actuator setting **706** as sent by application **171***i*, but both sets of data can achieve the same objective of having an actuator **101***y* apply a setting. According to one exemplary embodiment, actuator setting **706** as sent by module program **101***i* could be an analog voltage along a system bus **101***d*, while actuator setting **706** as sent by application **171***i* could be a string or number. Note that as contemplated herein, the term "actuator data" can include or comprise "actuator setting".

**[0276]** After applying actuator setting **706**, actuator **101***y* can send an acknowledgement to module program **101***i*. Module program **101***i* can then send a second message **208** to server **105**, where message **208** includes a server instruction **414** of "confirmation". The server instruction **414** of "confirmation" could be included in a module encrypted data **403** according to a preferred exemplary embodiment. Server **105** can receive the second message **208** with the module encrypted data **403** and decrypt the module encrypted data **403** using a step **413** to extract the server instruction **414** of "confirmation". The second message **208** may include the actuator identity **152**, a timestamp value **604***a*, and/or also the module identity **110**. Server **105** can send an application message **701** that includes a confirmation **705**, where the confirmation can (i) inform application **171***i* that the actuator setting **706** sent to server **105** has been properly and/or successfully applied by module **101** and/or actuator **101***y*. Confirmation **705** could also include module identity **110** and/or actuator identity **152** and a timestamp value **604***a*. Application **171***i* could then send an acknowledgement back to server **105** after receiving the confirmation **705**.

**[0277]** According to preferred exemplary embodiments, actuator identity **152** is preferably globally unique, such that that including an actuator identity **152** in any packet would allow a server **105** or application **171***i* to lookup a module identity **110** and/or module **101** using the actuator identity **152** and a database such as, but not limited to, module database **105***k*. Similarly, a sensor identity **151** may be globally unique, according to preferred exemplary embodiments such that a sensor identity **151** in any packet would allow a server **105** or application **171***i* to lookup a module identity **110** and/or module **101** using the sensor identity **151** and a database such as, but not limited to, application database **171***k*.

**[0278]** FIG. **8**

**[0279]** FIG. **8** is a simplified message flow diagram illustrating an exemplary system with exemplary data transferred between a module and an application using a server, in accordance with exemplary embodiments. System **800** can include an application server **171**, a server **105**, and a module **101** in connected via a network. The network could comprise the Internet **107**. Application server **171** could include an application **171***i*, where application **171***i* can include logic, algorithms, databases, user interfaces, and programs for managing a plurality of modules **101** with a plurality of users **183**. Application server **171** and application **171***i* may be associated with an M2M service provider **108**, and M2M service provider **108** could use application **171***i* to provide and manage a service with distributed modules **101** associated with a plurality of monitored units **119**. Server **105** could belong to a set of servers **105***n*, and the set of servers **105***n* could also take the actions described herein for the server **105**.

**[0280]** In an exemplary embodiment, module **101** can derive a public key **111** and a private key **112** using step **515**. Module **101** can derive the public and private keys using step **515** and a set of cryptographic parameters **126**. Alternatively, module **101** may have the module public key **111** and module private key **112** generated outside module **101** and loaded into a non-volatile memory **101***w*. Server **105** can utilize step **516** to receive a module public key **111** from module **101**. In an exemplary embodiment, module public key **111** in a step **516** illustrated in FIG. **8** can comprises a format of the module public key **111** that is different than a certificate **122** that could record a module public key **111**. Module public key **111** in a step **516** could be received in an exemplary format and form illustrated in FIG. **6***b* above.

**[0281]** Server **105** can utilize a step **517** to authenticate a message **208** that contains the module public key **111** and a module identity **110** received in a step **516**. Authentication of module public key **111** may be preferred in order to ensure that the module public key **111** is properly associated with the correct physical module **101** with a module identity **110**, and prevent an imposter, hacker, etc. from sending in a fake module public key **111** for module **101**. After using step **517** to authenticate module public key **111**, server **105** can record module public key **111** with the module identity **110** in a module database **105***k*, which could also comprise a shared module database **105***k* illustrated in FIG. **1***h*. Although not illustrated in FIG. **8**, server **105** can also send an application message **701** to application **171***i* after successfully recording an authenticated module public key **111**. Although not illustrated in FIG. **8**, a module public key **111** received in step **516** may also include a module public key identity **111***a* in order to track which of a plurality of potential module public keys **111** for a module **101** may be used at any given point in time or with any given message **208**.

**[0282]** Also, server **105** is not required to receive module public key **111** from module **101** in order to utilize the methods and systems contemplated herein. Instead of receiving module public key **111** in a message **208** from module **101**, server **105** could alternatively query another server such as, but not limited to, application server **171** or a server associated with certificate authority **118** for either module public key **111** or a certificate **122** associated with module **101** using a module identity **110**, where module identity **110** could be received in a message **208** at a step **516** with or without module public key **111**. In addition, server **105** could have a list or database table of module identities **110** and module public keys **111** loaded into a module database **105***k* before the message **208** in FIG. **8** is received. After recording module public key **111** and module identity **110**, possibly including a module public key identity **111***a*, server **105** can wait for wait interval **703**. Wait interval **703** could represent the time between reports or messages **208** submitted by module **101**, and wait interval **703** for an individual module **101** could comprise a wide range of values from several times a second to several days or longer, depending upon the application and/or monitored unit **119**. The wait interval **703** can end when server **105** receives a message **208** from module **101** with a module identity **110**.

**[0283]** Module controller **105***x* within server **105** can receive a message **208** that includes a server instruction **414** with sensor data **604***b*. The sensor data **604***b* and/or server instruction **414** could be included in a module encrypted data

**403**, where decrypting the module encrypted data **403** can use the module public key **111** submitted in step **516** above and derived by module **101** in step **515**. According to one exemplary embodiment, module encrypted data **403** could be ciphered with a symmetric key **127** that is a derived shared key **129***b* from a key derivation function **141***f* and module public key **111** received in step **516** (and also server public key **114**). Module controller **105***x* can process message **208** using the steps depicted and described in connection with FIG. **4** in order to decrypt the module encrypted data **403** and obtain the plaintext server instruction **414** and plaintext sensor data **604***b*. Although sensor data **604***b* is illustrated as the server instruction **414** in FIG. **8**, server instruction **414** could have other values such data associated with any of the components for module **101** illustrated in FIG. **1***b* and FIG. **1***e*. Server instruction **414** could be a "query" where module **101** queries for information from server **105** or application **171***i*, or server instruction **414** could be an alarm or error notification outside a regular reporting interval. Other possibilities for server instruction **414** exist without departing from the scope of the present invention. In an exemplary embodiment, server instruction **414** could also be a periodic "registration" message with no subsystem data for module **101** (an also no sensor data **604***b*), and a "registration" could be a message for server **105** indicating module **101** is awake and online with Internet **107**.

**[0284]** Server **105** can establish a secure connection with application server **171** and application **171***i* using a secure connection setup **801** and a secure connection data transfer **802**. Server **105** can utilize an application interface **105***i* to manage the communication with application **171***i* and/or application server **171**, while a module controller **105***x* can manage communication with a module **101**. Alternatively, application interface **105***i* and module controller **105***x* can be optionally combined or omitted, such that server **105** and/or a set of servers **105***n* perform the actions illustrated in FIG. **8** for application interface **105***i* and module controller **105***x*. Likewise, server **105** and application **171** could be combined or operate on the same local area network (LAN) and thus not be connected via the Internet **107**. If server **105** and application **171** are nodes within the same LAN or virtual private network (VPN), then the network connection can also be considered a secure connection (without using encryption between the nodes), since packets routed between the nodes may not need to traverse the Internet **107** and thus the network layer could provide security. Although secure connection setup **801** is illustrated in FIG. **8** as occurring after message **208** is received by server **105**, secure connection setup **801** could take place before message **208** is received by server **105**. Secure connection setup **801** could utilize a secure protocol such as, but not limited to, TLS, Secure Sockets Layer (SSL), IPSec, or VPN to establish a secure connection between server **105** and application **171***i* and/or application server **171**, such that data transferred between the two nodes is encrypted and also not subject to replay attacks. As contemplated herein, a secure connection can comprise any of a TLS connection, an IPSec connection, a VPN connection, an encrypted connection, and/or a LAN connection between server **105** and application server **171** and/or application **171***i*, and other possibilities exist as well without departing from the scope of the present invention.

**[0285]** Other secure connections may be utilized as well, including a secure shell (SSH) tunnel, future versions of standard secure connections, or also a proprietary protocol for a secure connection. Secure connection setup **801** as illustrated in FIG. **8** may utilize a TLS protocol, such as, but not limited to, TLS version 1.2, TLS version 1.3, etc. Secure connection setup **801** can include the transfer of a certificate **122** for application server **171**, and also the transfer of an application public key **171***w*. Server **105** can utilize application public key **171***w* to encrypt data received from module **101** in a message **208**, such as, but not limited to, sensor data **604***b*. According to one exemplary embodiment, application message **701** could be ciphered with a symmetric key **127** that comprises a derived shared key **129***b* from at least (i) a key derivation function **141***f*, (ii) application public key **171***w* and server public key **114**, and (iii) a random number **128***a* from either server **105** or application server **171**.

**[0286]** The message flow in a secure connection setup **801** also illustrates one benefit of the present invention, where a message **208** can be securely transferred between module **101** and server **105** using a single UDP datagram (or less than 3-4 datagrams), while secure connection setup **801** may require a plurality of TCP messages in both directions. In other words, using a secure connection setup **801** as illustrated in FIG. **8** between module **101** and server **105** may not be energy efficient for module **101**, while using secure connection setup **801** between server **105** and application server **171** can be efficient, since the data from a plurality of modules **101** can be shared over the secure connection setup **801**. Also note that since module **101** may sleep for relatively long periods such as, but not limited to, 30 minutes or longer, a new secure connection setup **801** would likely be required to support a firewall **104** after each period of sleep, and completing the process of a secure connection setup **801** each time module **101** wakes may not be energy or bandwidth efficient for a module **101**.

**[0287]** After completing server connection setup **801**, in exemplary embodiments server **105** or a set of servers **105** can use a secure connection data transfer **802** to send a first application message **701**, where the first application message **701** could include update instruction **704** that includes sensor data **604***b* that server **105** received in a message **208**. Data within the first application message **701** containing update instruction **704** could be ciphered according to the specifications of the secure connection, such as, but not limited to, TLS or IPSec, and other possibilities exist as well. Note that server **105** can decrypt a module encrypted data **403** that includes sensor data **604***b* and subsequently encrypt the sensor data **604***b* according to the format required by secure connection setup **801** for transfer to application **171***i* using secure connection data transfer **802**. System **700** can use two different server public keys **114**, recorded in the form of a certificate **122** in one embodiment, with a first server public key **114** used in encrypting and/or decryption module encrypted data **403** and a second server public key **114** used in encrypting and/or decrypting update instruction **704**. The two server public keys **114** can be used by server **105** in a key derivation function **141***f* to derive two shared secret keys **129***b* used in a symmetric ciphering algorithm **141***b* for both secure connection data transfer **802** and module encrypted data **403** (with a different derived shared public key **129***b* with module **101** and application server **171**, respectively).

**[0288]** In another embodiment, server **105** can use the same server public key **114** to both decrypt module encrypted data **403** and encrypt update instruction **704**. Other possibilities exist as well for server **105** to use a server public key **114** to (i) encrypt update instruction **704**, such as using an asymmetric

ciphering algorithm **141***a*, and (ii) decrypt module encrypted data **403** without departing from the scope of the present invention. As illustrated in FIG. **8**, server **105** can receive an acknowledgement **804** after sending the first application message **701**, with update instruction **704** that includes sensor data **604***b*, where acknowledgement **804** can signal that application message **701** with update instruction **704** has been received by application **171***i* and/or application server **171**. Although not illustrated in FIG. **8**, the acknowledgement **804** could optionally include a module instruction **502** for module **101**. As contemplated herein, a module instruction **502** can also include a timestamp value **604***a*, such that a module **101** can determine when the module instruction **502** was generated or processed by a source of the module instruction **502** such as an application **171***i* or a server **105**.

**[0289]** After receiving message **208**, server **105** can then send a response **209**. Response **209** could be sent before or after server **105** sends update instruction **704** to application **171***i* using secure connection data transfer **802**. Response **209** can include a server encrypted data **504** that includes a module instruction **502**. Module instruction **502** could be (i) processed by server **105**, (ii) obtained by server **105** from application **171***i* in an application instruction **701**, and/or (iii) read by server **105** from a shared module database **105***k*. In other words, a secure connection data transfer **802** may be utilized by a server **105** and either (i) an application server **171** or (ii) a shared module database **105***k* to in order for server **105** or a set of servers **105***n* to receive a module instruction **502** with a module identity **110** for a module **101**. According to an exemplary preferred embodiment, server **105** waits for a response or acknowledgement **804** from application **171***i* to application message **701** (where application message **701** could comprise a polling request **1302** described below) before sending response **209** to module **101**. One reason for waiting for a response or acknowledgement **804** from application **171***i* is that response or acknowledgement **804** from application **171***i* could include a module instruction **502**, and the module instruction **502** may preferably be included in a response **209**. Other possibilities exist as well without departing from the scope of the present invention.

**[0290]** FIG. **8** can also illustrate a benefit of an exemplary embodiment contemplated herein. According to an exemplary embodiment, (i) server **105** and application server **171** can utilize a first set of cryptographic algorithms **141** for sending and receiving data between server **105** and application server **171**, such as, but not limited to, with a secure connection data transfer **802**, and (ii) server **105** and module **101** can utilize a second set of cryptographic algorithms **141** for sending and receiving data between server **105** and module **101**, such as using the second set of cryptographic algorithms **141** for a module encrypted data **403** and/or server encrypted data **504**. In an exemplary embodiment, server **105** and application server **171** can use RSA algorithms **153** in the first set of cryptographic algorithms **141**, while server **105** and module **101** can use ECC algorithms **154** in the second set of cryptographic algorithms **141**. As one example, server **105** can use an (i) RSA-based asymmetric ciphering algorithm **141***b* and first server public key **114** with the application server **171** to securely transfer a first symmetric key **127** with application server **171**, and (ii) an ECC-based asymmetric ciphering algorithm **141***b* and second server public key **114** with the module **101** to securely transfer a second symmetric key **127** with a module **101**.

**[0291]** Other possibilities exist as well for a server **105** to use a different cryptographic algorithms **141** and/or cryptographic parameters **126** for each of application server **171** and module **101**. (A) Server **105** and application server **171** could use a first set of cryptographic parameters **126** for use with cryptographic algorithms **141** for an application message **701** with related server digital signatures, while (B) server **105** and module **101** could use a second set of cryptographic parameters **126** for use with cryptographic algorithms **141** for a module encrypted data **403** and/or server encrypted data **504** and related digital signatures. The first set of cryptographic parameters **126** and the second set of cryptographic parameters **126** are illustrated in FIG. **8**. In order to maximize security between servers such as server **105** and application server **171**, the first set of parameters **126** could specify (i) a longer public and private key length, (ii) a shorter key expiration time **133**, (iii) a longer secure hash algorithm (such as, but not limited to, an exemplary 384 or 512 bits), (iv) a longer symmetric ciphering key **127** length (such as, but not limited to, an exemplary 192 or 256 bits), (v) the use of or values for RSA algorithm **153** and a modulus, (vi) the use of Diffie Hellman key exchange or a first key exchange algorithm for a key derivation function **141***f* and key exchange, (vii) the use of or values for a second symmetric ciphering algorithm **141***b* for symmetric ciphering, (viii) the use of or values for an RSA digital signature algorithm or a second digital signature algorithm, and similar settings.

**[0292]** In accordance with a preferred exemplary embodiment, in order to minimize processing power and/or energy usage required for a module **101**, the second set of cryptographic parameters **126** illustrated in FIG. **8** could specify (i) a shorter public and private key length, (ii) a longer key expiration time **133**, (iii) a shorter secure hash algorithm (such as, but not limited to, an exemplary 224, 256, or 160 bits), (iv) a shorter symmetric ciphering key **127** length (such as, but not limited to, an exemplary 128 bits), and (v) the use of an ECC algorithm **154**, (vi) the use of or values for an ECC standard curve **138** and/or ECC parameters **137**, (vii) the use of or values for ECDH **159** or a second key exchange algorithm for key derivation and exchange, (vii) the use of or values for of a second symmetric ciphering algorithm **141***b* for symmetric ciphering, (viii) the use of or values for of ECDSA **158** or a second digital signature algorithm for digital signatures, and similar settings. In an embodiment, the first set of parameters **126** (which can be used by server **105** and application server **171**) and the second set of parameters **126** (which can be used by server **105** and module **101**) can both specify the use of elliptic curve cryptographic algorithms **141**, but with different sets of parameters **126** such that the first set of parameters **126** is selected for server to server communications, and the second set of parameters **126** is selected for communications between a server **105** and a module **101**. In another embodiment, the first set of parameters **126** and the second set of parameters **126** can both specify the use of RSA based cryptographic algorithms **141**, but with different sets of parameters **126** such that the first set of parameters **126** is selected for server to server communications, and the second set of parameters **126** is selected for communications between a server **105** and a module **101**.

**[0293]** In this manner, the use of cryptographic algorithms **141** between (i) server **105** and application server **171** and (ii) server **105** and module **101** can be optimized given different constraints for processing power and energy consumption for server **105**, application server **171**, and a module **101**. In

addition, an application server **171** may use cryptographic algorithms **141** and parameters **126** that may not be compatible with cryptographic algorithms **141** and parameters **126** used by a module **101**, and server **105** can use cryptographic algorithms **141** and at least the two sets of cryptographic parameters **126** illustrated in FIG. **8** to enable a translation or conversion of encrypted data and digital signatures between a module **101** and an application server **171**, thereby establishing connectivity between a module **101** and an application server **171** through a server **105**. According to an exemplary embodiment, server **105** can function as a gateway between application server **171** and/or application **171***i* and a plurality of modules **101**.

**[0294]** FIG. **9**

**[0295]** FIG. **9** is a simplified message flow diagram illustrating exemplary data transferred between (i) a server and an application and between (ii) a server and a module, in accordance with exemplary embodiments. An application server **171**, a server **105**, and a module **101** can send and receive data illustrated in FIG. **9**. Application server **171** can include application **171***i* and use an Internet Protocol address and port (IP:port) number **903** for sending and receiving data with server **105**. Server **105** can include an application interface **105***i* and a module controller **105***x*, where application interface **105***i* can access a first server IP:port number **901** for communicating with application server **171**, and module controller **105***x* can access a second server IP:port number **207** for communication with module **101**. In accordance with a preferred exemplary embodiment, multiple modules **101** can send data to server IP:port number **207**, and thus server **105** and/or a module controller **105***x* can use a single IP:port number **207** to communicate with a plurality of modules **101**. In addition, server **105** could specify that one subset of modules **101** communicate with a first IP:port number **207**, and a second subset of modules **101** communicate with a second IP:port number **207**, etc. In another embodiment, a set of servers **105***n* could comprise the server **105** illustrated in FIG. **8**. Module **101** can utilize an IP:port number **204** for sending and receiving data with a server **105**.

**[0296]** As illustrated in FIG. **9**, a symmetric firewall **104** could be included between module **101** and server **105**. The of IP addresses and port numbers in packets between server **105** and module **101** illustrated in FIG. **9** could also represent routing if a firewall **104** is present and functions as a symmetric firewall without NAT routing. In this case, firewall **104** may not perform network address translation on source and destination IP addresses and/or port numbers, but rather filter packets based on pre-determined rules. For example, a firewall **104** that is a symmetric firewall could drop inbound packets from IP:port number **207** to module **101** unless module **101** had previously sent a packet to IP:port number **207** within a firewall port binding timeout value **117**. Alternatively, a firewall **104** may be optionally omitted, and in this case the destination address in packets sent from server **105** to module **101** could include the IP address **202** of module **101**, which is also the case illustrated in FIG. **9**. In other words, FIG. **9** illustrates an exemplary routing of packets in the cases that (i) firewall **104** is a symmetric firewall, and (ii) firewall **104** is optionally not present.

**[0297]** Server **105** can receive a message **208** from a module **101**. Server **105** can use a module controller **105***x* to receive the message, and module controller **105***x* could also be identified as a process operating on server **105** that binds to the port number in IP:port **207**, which could include a port number **205**. Message **208** could include module identity string **904**, which could represent a temporary or transient string or number used by module **101** and server **105** to associate and identify message **208** with module identity **110**. Module identity string **904** could also comprise a module identity **110**. Server **105** can use module identity string **904** to select a symmetric key **127** in order to decrypt module encrypted data **403**, since module identity string **904** may preferably be not encrypted. Server **105** and module **101** could use an algorithm within cryptographic algorithms **141** in order to process a module identity string **904**, whereby the module identity string **904** can be converted between (i) a module identity **110** in a form such as, but not limited to, a serial number, IMEI, or related identifier for module **101**, and (ii) a module identity string **904** in a message **208** that can traverse the Internet **107**.

**[0298]** Message **208** as received by server **105** can also include a server instruction **414** within a module encrypted data **403**, where the module encrypted data **403** could be ciphered using a symmetric key **127**. The server instruction **414** illustrated in FIG. **9** can be an exemplary "update" instruction, where the "update" instruction can include a security token **401** and sensor data **604***b*. Sensor data **604***b* can include a sensor identity **151** and a sensor measurement. Server instruction **414** within a message **208** could include many other values besides an update, including a registration, a query, an alarm or error notification, configuration request, software request, confirmation, or other values also depicted and described in connection with a server instruction **414** in FIG. **4**. A security token **401** can comprise a random number **128***a* processed by a random number generator **128** and can be preferably not reused and therefore can keep message **208** unique and not subject to replay attacks. In exemplary embodiments, a UDP protocol may be implemented for message **208**, and the connectionless UDP protocol may require a module **101** to send retransmissions of a UDP datagram **601***a* for message **208**, if module **101** does not receive a response **209** within a specified timer period.

**[0299]** If the UDP Lite protocol is utilized for message **208**, with multiple copies of UDP Lite datagram **601***a* received in an exemplary embodiment, then each UDP Lite datagram **601***a* could be different, depending on the presence of bit errors in the datagram, and thus server **105** can use timer **905** to collect the multiple copies of UDP Lite datagram **601***a* within the timer **905** period and process the multiple packets received, including combining the data across multiple packets, in order to eliminate bit errors within the datagrams and collect an error-free message **208**. Packets for a message **208** received outside timer **905** could be dropped by server **105**, and the timer **905** could start when the first datagram **601***a* for a message **208** was received by server **105**.

**[0300]** After receiving message **208**, server **105** use the steps outlined in FIG. **5***a* to process message **208** and read the plaintext server instruction **414**, such as, but not limited to, the sensor data **604***b* illustrated in FIG. **9**. Other possibilities exist as well for sensor data **604***b* or values or information inside a server instruction **414**. Server **105** can then send or transmit a first application message **701** to application server **171** that includes data received from the server instruction **414** from module **101** in message **208**. The data received in the server instruction **414** from module **101** could be included by server **105** in an update instruction **704**. An application **171***i* operating within application server **171** or associated with application server **171** could receive the first application message

**701**. The first application message **701** could be formatted according to a TCP datagram **902**, although other possibilities exist as well including UDP.

**[0301]** In accordance with an exemplary preferred embodiment, the first application message **701** may include an update instruction **704** with sensor data **604***b*, although update instruction **704** could also contain or include other data pertaining to module **101** besides sensor data **604***b*, such as a state of a component with module **101**, a state of a software routine, variable, or parameter associated with module **101**. The first application message **701** sent from server **105** to application server **171** could be a datagram within a secure connection data transfer **802** as illustrated in FIG. **8**. Sensor data **604***b* could be sent by server **105** using application server public key **171***w*, such as either (i) mutually deriving a common shared key **129***b* between server **105** and application **171***i* using a key derivation function **141***f*, where the shared key **129***b* could function as a symmetric key **127** with a symmetric ciphering algorithm **141***b*, or (ii) server **105** sending a symmetric key **127** to application server **171** using an asymmetric ciphering algorithm **141***a* and the application server public key **171***w*. Message **805** in FIG. **8** with the label of "Client Key Exchange" can comprise server **105** sending a symmetric key **127** (or value or cryptographic parameters **126** for deriving symmetric key **127**) to application server **171**, where the symmetric key **127** can be used by server **105** to encrypt update instruction **704** illustrated in FIG. **9**. As contemplated herein, a random number **128***a* input into a set of cryptographic algorithms **141***d* can also be considered a cryptographic parameter **126**. Also, a random number **128***a* input into a key derivation function **141***f* can comprise a cryptographic parameter **126**.

**[0302]** In accordance with an exemplary preferred embodiment, application message **701** may include (i) module identity **110** encrypted within secure connection data transfer **802** and also a server identity **206** that is not encrypted. In this manner, application server **171** can use server identity **206** to select a symmetric key **127** (possibly sent in message **805** as described in the paragraph above) in order to decrypt the encrypted data in update instruction **704**. Application server **171** can receive the first application message **701** sent by server **105** and process the message. The message processing by application server **171** could use steps similar or equivalent to the steps utilized by server **105** illustrated in FIG. **4**, in order to extract a plaintext application instruction **704**. Although not illustrated in FIG. **9**, application **171***i* could record data received within application instruction **704** and record the data in an application database **171***k*. Application **171***i* could use the data received in application instruction **704** or a plurality of application instructions **704** to generate reports, graphs, emails, or other user information for a user **183**.

**[0303]** Upon processing the information within application instruction **704**, application **171***i* or application server **171** could send a second application message **701** to server **105**, as illustrated in FIG. **9**. The second application message **701** could be sent using a secure connection data transfer **802**, and could include a module instruction **502** and a module identity **110**. The second application message **701** can use the IP:port number **903** as a source IP:port number for the second application message **701**, where IP:port number **903** also represented a destination IP:port number for the first application message **701**. The second application message **701** can use the IP:port number **901** as the destination IP:port number, where IP:port number **901** was the source port number in the first application message **701**. The module instruction **502** within the second application message **701** could include an actuator setting **706**. The module instruction **502** within the second application message **701** can comprise other data or module instructions **502** for a module **101** that do not include an actuator setting **706**, such as the exemplary data depicted and described in connection with FIG. **5***a*.

**[0304]** Server **105** or a set of servers **105***n* can receive the second application message **701**, and the message could be received using an IP:port number **901**. Although an IPv4 address is shown in FIG. **9**, and IPv6 address could be utilized as well. Server **105** could decrypt a body **602**, that contains module identity **110** and a module instruction **502**, using algorithms specified according to a secure connection data transfer **802**. As depicted and described in FIG. **8**, a first set of cryptographic parameters **126** with cryptographic algorithms **141** could be used with an application message **701** and a second set of cryptographic parameters **126** with cryptographic algorithms **141** could be used with server encrypted data **504** and/or module encrypted data **403**.

**[0305]** After extracting a plaintext module instruction **502** and module identity **110** from a body **602** in the second application message **701**, server **105** can take steps to process the data and create a response **209** for module **101**. Server **105** can record or query for information pertaining to module **101** using module identity **110** in a module database **105***k*. In accordance with exemplary embodiments, server **105** can use module identity **110** received in the second application message **701** to select (i) a symmetric key **127** used by module **101** for encrypting and/or decrypting a server encrypted data **504** that can include the module instruction **502**, (ii) a destination IP:port number **204** for sending a response **209**, (iii) a source IP:port number **207** for sending a response **209**, (iv) a determination if a wait interval **703** is required before sending response **209**, (v) a value for a security token **401**, and (vi) at least one value for a set of cryptographic parameters **126** for use with a cryptographic algorithms **141** in communications with module **101**. In one embodiment, different modules **101** connected to server **105** may use different cryptographic parameters **126**, and server **105** can select the appropriate set of cryptographic parameters **126** for a module **101** using (a) the module identity **110** received in the second application message and (b) a module database **105***k*. Server **105** can also use module identity **110** received in the second application message **701** to select (vii) a transport protocol for a response **209**, such as, but not limited to, TCP, UDP, or UDP Light, and (viii) a channel coding **406** parameter such as, but not limited to, a block code, turbo code, or forward error correction coding scheme. Server **105** can use module identity **110** received in an application message **701** such as the second application message **701** illustrated in FIG. **9** to format and/or send a response **209** to module **101**.

**[0306]** According to a preferred exemplary embodiment, server **105** may receive an application message **701** with data for a module **101** at arbitrary times. According to a preferred exemplary embodiment, server **105** can use module identity **110** received within an application message **701** to determine (i) if server **105** should wait until a wait interval **703** expires before sending response **209** (where the wait interval **703** can end upon receipt of a message **208** from a module **101** with the module identity **110** received in the application message **701**) or (ii) if server **105** can send response **209** right away (such as a firewall port binding timeout period **117** has not expired), where response **209** includes the module instruction

**502** received in the application message **701**. Firewall port binding timeout value **117** (or a time value associated with firewall port binding timeout value) can be recorded for module identity **110** in a module database **105***k*.

**[0307]** After (A) using module identity **110** received within application message **701** to select values to process a response **209** and timing for sending a response **209**, then (B) server **105** can send response **209** as illustrated in FIG. **9**, where the specific response **209** in FIG. **9** is exemplary. Response **209** can include a server encrypted data **504**. Server encrypted data **504** can include module instruction **502**. The exemplary response **209** illustrated in FIG. **9** includes an actuator setting **706** within module instruction **502**, but other possibilities exist as well. Note that the use of server encrypted data **504** is optional within a response **209**, and server **105** could send module instruction **502** as plaintext. However, in this case of module instruction **502** being sent as plaintext, server **105** can preferably include a server digital signature **506** such that module **101** can verify the server digital signature **506** using the server public key **114** and confirm the module instruction **502** was transmitted by server **105**. In accordance with exemplary preferred embodiments, (i) a message from module **101** to server **105** that does not include a module encrypted data **403** preferably includes a module digital signature **405**, and (ii) a response **209**, message sent back, datagram, or packet from server **105** to module **101** that does not include a server encrypted data **504** preferably includes a server digital signature **506**. If data is not encrypted within a packet and the packet includes plaintext instructions such as a module instruction **502** or a server instruction **414**, then, in accordance with preferred exemplary embodiments, the receiving node can preferably verify the identity of a sender using (i) a digital signature, (ii) an identity, and (iii) a public key, where the digital signature and identity can be included in the packet.

**[0308]** Response **209** sent from server **105** to module **101** could include a checksum **603**. Since firewall **104** may comprise a symmetric firewall **104** (that may not perform network address translation routing), the destination address within IP:port **204** in response **209** illustrated in FIG. **9** may match the IP address **202** used by module **101**. In this case, where the destination IP:port in response **209** includes IP address **202**, a checksum **603** sent by server **105** can be equal to a checksum **603** received by module **101**. In accordance with exemplary embodiments, response **209** is transmitted or sent by server **105** within a firewall port binding timeout value **117** after message **208** was received by server **105**. In other words, if a firewall port binding timeout value **117** was equal to an exemplary 20 seconds for UDP packets, the response **209** illustrated in FIG. **9** would preferably be sent in less than 20 seconds after receiving the previous message **208**.

**[0309]** FIG. **10**

**[0310]** FIG. **10** is a flow chart illustrating exemplary steps for a set of servers to communicate with a module, in accordance with exemplary embodiments. The exemplary steps illustrated in FIG. **10** could be implemented in either a collection of servers **105** (such as, but not limited to, the two exemplary servers illustrated in FIG. **1***h*), where the collection of servers **105** comprise a set of servers **105***n*. Or, a set of servers **105***n* can comprise a single server **105** with only one member in the set of servers **105**. The members and numbers of servers **105** in a set of servers **105***n* can also change over time. In other words, over time such as when a plurality of module public keys **111** could be generated for various needs of a module **101** or a system such as, but not limited to, system **100**, module **101** may communicate with multiple servers in some embodiments.

**[0311]** FIG. **10** illustrates an exemplary embodiment where a set of servers **105***n* can authenticate module **101** using a module identity **110** and subsequently receive a plurality of module public keys **111** over time. As depicted and described in FIG. **1***h*, a set of servers **105***n* can comprise at least on server **105**. New module public keys **111** are generated for the various purposes contemplated herein, including (i) periodically rotating module private keys **112** for security, (ii) changing a set of cryptographic parameters **126** used with the keys in order to increase security (where a new set of cryptographic parameters **126** can require the use of a new module public key **111** and new module private key **112**), (iii) change of ownership and/or control of module **101** such that the previous module private key **112** may not longer be considered secure, and (iv) the first time module **101** sends in a module public key **111**. Other possibilities for reasons that a set of servers **105***n* can receive and authenticate and/or verify a module public key **111** are possible as well without departing from the scope of the present invention.

**[0312]** At step **1001**, a server **105** and/or a set of servers **105***n* can receive and verify a module public key **111** is associated with a module identity **110** that is recorded within server **105**, potentially in a module database **105***k*. Module database **105***k* could also be a shared module database **105***k* as illustrated in FIG. **1***h*. The module public key **111** could be received from module **101** in a message **208** that includes the module identity **110**. If a server **105** has not previously record module identity **110** received in a message **208** at step **1001**, potentially in a module database **105***k*, then server **105** could query for data to authenticate module public key **111** with module identity at step **1001**. A server **105** could query other servers such as, but not limited to, an application server **171**, a certificate authority **118**, and/or a server associated with M2M service provider **108** or module provider **109**. The other exemplary servers listed in the previous sentence could also comprise members of a set of servers **105***n* in some embodiments, but in other embodiments an application server **171** and a certificate authority **118**, etc. may not be members of the set of servers **105**. Exemplary details for the steps to verify a received module public key **111** are also depicted and described in connection with step **1202** of FIG. **12**, and step **517** of FIG. **5***b*. In accordance with an exemplary embodiment, the received module public key **111** can be verified using any of a shared secret key **510**, a module digital signature **405**, or a certificate **122**. A server **105** could also use an initial set of cryptographic parameters **126** at step **1001**, where the initial set or first set of cryptographic parameters **126** could be pre-agreed between module **101** (possibly through module provider **109**) and server **105** (possibly through M2M service provider **108**).

**[0313]** According to an exemplary preferred embodiment, (i) the first time a server **105**, including any server in a set of servers **105**, receives any module public key **111** for module identity **110**, the module public key **111** can be verified using a certificate **122**, and (ii) a subsequent time server **105** receives a module public key **111** for module identity **110**, the module public key **111** can be verified using either a shared secret key **510** or a module digital signature **405**, where (i) the module digital signature **405** is processed by server **105** using a prior module public key **111** (i.e. received before step **1001**), and (ii) the prior module public key **111** had also been previ-

ously verified. In the embodiment where a received module public key **111** at step **1001** is verified using a prior module public key **111** and a module digital signature **405** (as contemplated in the previous sentence), a message **208** including the module digital signature **405** may also preferably include a module public key identity **111***a* such that server **105** can properly lookup, query, or obtain the correct prior module public key **111** with module public key identity **111***a* to use with a digital signature algorithms **141***d* to verify the module digital signature **405** received in the message **208**. In other words, when a plurality of module public keys **111** may be utilized, server **105**, possibly within a set of servers **105***n*, can use a module public key identity **111***a* to track which module public key **111** is currently being used with either a module digital signature **405** or an asymmetric ciphering algorithms **141***a*.

**[0314]** After receiving and verifying module public key **111** and module identity **110** at step **1001**, a server **105** and/or a set of servers **105***n* can receive a message **208** that includes module identity **110** at step **1002**. The message **208** could include a server instruction **414** or a module encrypted data **403**. In an exemplary embodiment, server **105** can receive other messages **208** and module public keys **111** both before and after steps **1001** and step **1002**, as well as other steps contemplated herein. In other words, the various messages and responses illustrated in Figures herein can comprise subsets of all messages and responses, such that the subsets comprise embodiments of the present invention. At step **1003**, server **105** can send a response **209**, where the response can include a second set of cryptographic parameters **126**. The response **209** can be sent in a packet with a source IP:port number and a destination IP:port number, and the destination IP:port number in the packet can be equal to or the same as the source IP:port number for a packet received in message **208** at step **1002**.

**[0315]** In an exemplary embodiment, the second set of cryptographic parameters **126** are sent to a module **101** with module identity **110** only after the module public key **111** has been verified in a step **1001**. In this manner, the cryptographic parameters **126** may be more securely held (i.e. not disclosed to unauthorized parties). Further, the cryptographic parameters **126** in a response **209** sent at step **1003** may also optionally be encrypted using the module public key **111** received at step **1001**. In one embodiment, the module public key **111** received in step **1001** can be used to derive or transfer a symmetric key **127**, and the symmetric key **127** could be used with a symmetric ciphering algorithms **141***b* to cipher the second cryptographic parameters **126** sent in a response **126**.

**[0316]** At step **1004**, a set of servers **105***n* can receive over time a series of module public keys **111** associated with module **101** using module identity **110**. Members of the series of module public keys **111** can be different, representing different module public keys **111** for the module identity **110**, and the numbers and/or strings in the module public keys **111** can be different. The series of different module public keys **111** can comprise at least a first module public key **111** for module identity **110** and a second module public key **111** for module identity **110**, where the two module public keys **111** are received at different times, such as, but not limited to, exemplary values of a week, a month, or a year apart, and other times between members of the series of module public keys are possible as well. An exemplary format for a server **105** to receive a module public key **111** is illustrated in the exemplary message **208** depicted and described in connection with FIG. **6***b*, and other possibilities exist as well. Different members of the set of servers **105***n* can receive different module public keys **111** in the series of module public keys **111** for the module identity **110**. Although not illustrated in FIG. **10**, the set of servers **105***n* can also receive additional messages **208** and send additional responses **209** during step **1004**, such as when module **101** and the set of servers **105***n* continue to operate over time until step **1005** below. The module identity **110** received with a first module public key **111** in the series may have a different value, string, or number than the module identity **110** received with a second module public key **111** in the series, but different values, strings, or numbers for module identity **110** can used for the same physical module **101**, and the different values for module identity **110** can be associated with a unique serial number for a module **101**, and other possibilities exist as well.

**[0317]** A module **101** could generate, process, or derive each of the different module public keys **111** in the series of different module public keys **111** using a set of cryptographic algorithms **141**, the cryptographic parameters **126** sent at step **1003**, and a random number generator **128**. Each member of the series of module public keys **111** could be received in a message **208** that could also include a module public key identity **111***a* in order to track the module public keys **111**. In an exemplary embodiment, a server **105** at step **1004** can also receive a third set of cryptographic parameters **126** from module **101**, such that the third set of cryptographic parameters **126** received can specify how server **105** can use a set of cryptographic algorithms **141** in order to either (i) use at least one module public key **111** in the series form step **1004**, and/or (ii) communicate with module **101**. The third set of cryptographic parameters **126** could be sent in a module encrypted data **403**. Note that the second set of cryptographic parameters **126** sent by a server **105** at step **1003** could intersect with a third set of cryptographic parameters **126** received by server **105** with a module public key **111** at step **1004**.

**[0318]** In an exemplary embodiment, a second set of cryptographic parameters **126** sent by a server **105** at step **1003** could include a list of secure hash algorithms, and elliptic curve names, and a third set of cryptographic parameters received by server **105** in a step **1004** can include a selection by module **101** of a specific secure hash algorithm and an elliptic curve name from the first set of cryptographic parameters. Other possibilities exist as well, and each of the second set and third set of cryptographic parameters **126** can include more than a list of secure hash algorithms and elliptic curve names, such as but not limited to (i) the name or value for a symmetric ciphering algorithm **141***b*, (ii) parameters or values for a module random seed file **139**, (iii) the name or value for an asymmetric ciphering algorithm **141***a*, (iv) the name or value for a digital signature algorithm **141***d*, (iv) a value for a key pair generation algorithm, and/or (v) a value for a key derivation function **141***f*. The selection of the third set of cryptographic parameters **126** by module **101** could be made based on the capabilities of cryptographic algorithms **141** in a module **101**. In an exemplary embodiment, the third set of cryptographic parameters **126** received by server **105** at step **1004** comprises a subset of the second cryptographic parameters **126** sent by server **105** at step **1003**. After receiving the second cryptographic parameters at step **1004**, server **105** can record and implement the third set of cryptographic parameters **126** in future communications with module **101** (until possibly a different or new third set of cryptographic parameters **126** are possibly received by a server **105** in a message

**208** at a future time). Note that the use of a third set of cryptographic parameters **126** at step **1004** may optionally be omitted (as illustrated in FIG. **10**), such that the second set of cryptographic parameters **126** from step **1003** are used by module **101** at step **1004**.

**[0319]** At step **1005**, a server **105**, possibly in the set of servers **105**, can receive a module instruction **502** and a module identity **110**. In one embodiment, server **105** could poll another server, process, or database in order to receive the module instruction **502** and module identity **110**, such as, but not limited to, sending a polling request or query in a step **1302** depicted and described below in connection with FIG. **13** and FIG. **14**. The response received to the poll or query could be the receipt of a module instruction **502** and module identity **110** in a step **1005**, possibly through using a step **1303** illustrated in FIG. **13** and FIG. **14**. In another embodiment for step **1005**, a server **105** could receive an application message **701** from an application server **171**, where the application message **701** can include the module instruction **502** and the module identity **110**. The module instruction **502** for module **101** with module identity **110** could be for any reason that application **171***i* and/or user **183** prefers to change a state of module **101**, including the exemplary reasons depicted and described in connection with FIG. **7**. By receiving module instruction **502**, server **105** can enable the remote or external control of a module **101**, which may be important for successful operation of module **101**. At step **1005**, server **105** can record module instruction **502** in memory **105***e* or a module database **105***k*.

**[0320]** Although not illustrated in FIG. **10**, for the embodiment where server **105** receives a module instruction **502** in an application message **701** from application server **171**, server **105** could then use a wait interval **703**, to wait for the next message **208** from module **101**. A wait interval **703** as depicted and described in connection with FIG. **7**, and server **105** can wait after step **1005** and before step **1006**, or until a next message **208** is received with the module identity **110**. In many embodiments, module **101** may not be continuously connected with server **105** due to any of (i) the use of sleep or dormant states, (ii) periodic outages of network connectivity through a network **102** and/or the Internet **107**, and/or (iii) firewall rules on a firewall **104** that would prevent outbound packets from server **105** from reaching module **101**. Although not illustrated in FIG. **10**, server **105** could attempt to send a packet such as datagram **601***b* to module **101** at step **703** in FIG. **10**, and if module **101** does not send back a message **208** with a server instruction **414** of a confirmation and/or acknowledgement (potentially for the reasons listed in the above sentence), then server **105** could also then continue to wait using a wait interval **703**.

**[0321]** At step **1006**, server **105** can receive the next message **208** from module **101**, where message **208** preferably includes the module identity **110** and the module identity **110** can correspond to the module identity **110** received at steps **1005**, **1004**, and **1001**. The next message **208** illustrated in FIG. **10** at step **1006** could be for any reason. Server **105** can use the receipt of next message **208** at step **1006** as confirmation that module **101** is in an active state and that communication is possible through the Internet **107**, firewall **104**, and network **102**. The next message **208** at step **1006** may preferably include at least one of (i) module encrypted data **403** that is ciphered with a symmetric key **127** and (ii) a module digital signature **405**. In this manner, server **105** can verify or confirm that the next message **208** at step **1006** is from a module **101** with a module identity **110**.

**[0322]** At step **1007**, server **105** can send a second response **209** that includes the module instruction **502**, where response **209** can be sent to a module **101** with module identity **110**, and response **209** can be sent in after receiving the next message from step **1006**. Note that the second response **209** should preferably be sent before the expiration of a firewall port-binding timeout value **117**. The second response **209** could include server encrypted data **504**, where the module instruction **502** is included in the server encrypted data **504**. Alternatively, module instruction **502** could be sent a plaintext in the second response **209**, and in this case the second response **209** can preferably include a server digital signature **506**. Although not illustrated in FIG. **10**, after sending the second response **209** using a step **1007** illustrated in FIG. **10**, the set of servers **105***n* (of which a server **105** can be a member) can then also receive a confirmation with a timestamp **604***a* from module **101** with module identity **110**. The set of servers **105***n* could then send the timestamp **604***a* and module identity **110** to an application server **171** that originated the module instruction **502**, thereby informing the application server **171** when the module **101** executed the module instruction **502**. Other possibilities exist as well to those of ordinary skill in the art without departing from the scope of the present invention.

**[0323]** FIG. **11**

**[0324]** FIG. **11** is a flow chart illustrating exemplary steps for a set of servers to communicate with a module and an application server, in accordance with exemplary embodiments. The exemplary steps illustrated in FIG. **11** could be implemented in either a collection of servers **105** (such as, but not limited to, the two servers **105** illustrated in FIG. **1***h*), where the collection of servers comprise a set of servers **105***n*. Or, a set of servers can comprise a single server **105** with only one member in the set of servers **105***n*. In an exemplary embodiment, a set of servers **105***n* could derive the set of server's own server public key **114** and server private key **105***c* using a set of cryptographic algorithms **141**, a random number generator **128**, and a set of cryptographic parameters **126**. The set of servers **105***n* could use steps similar to step **515** for a module **101** in order to derive one or more server private keys **105***c*. According to an exemplary preferred embodiment, a set of servers **105***n* can use a plurality of public and private key pairs in order to efficiently and securely communicate through systems such as, but not limited to, those illustrated in system **100**, system **199**, system **700**, system **800**, system **1200**, and/or system **1300**.

**[0325]** A server public key **114** could be recorded in the form of a certificate **122** an optionally signed by a certificate authority **118**, and the certificate **122** may also optionally include a set of cryptographic parameters **126** associated with a server public key **114**. In an embodiment, a certificate **122** can include a subset of the set of cryptographic parameters **126** associated with the server public key **114**, and other members of the set outside the subset can be sent to a module **101** in a server encrypted data **503**. In one embodiment, a server public key **114** is kept confidential and not shared with other entities besides a set of modules **101** and/or application server **171**. In an exemplary embodiment, the server public key **114** is only transmitted to the set of modules **101** within a server encrypted data **503**, in order to increase the security of a system contemplated herein. Different pairs of keys within a plurality of public and private key pairs for a set of servers **105***n* can utilize different sets of cryptographic parameters

**126**. An exemplary use for a set of servers **105***n* using different pairs of server public key **114** and server private key **105***c* with different parameters **126** is illustrated in FIG. **11**, and other possibilities for the use of multiple pairs of public and private keys are possible as well without departing from the scope of the present invention.

**[0326]** At step **1101**, in an exemplary embodiment a set of servers **105***n* can establish a secure connection with at least one application server **171** using a first server private key **105***c* and a first set of cryptographic parameters **126**. The secure connection in step **1101** could be established through a secure connection setup **801** illustrated in FIG. **8**. The first set of cryptographic parameters **126** could specify multiple values across a set of algorithms comprising (i) asymmetric ciphering algorithms **141***a*, (ii) symmetric ciphering **141***b* algorithms, (iii) secure hash algorithms **141***c*, (iv) digital signature algorithms **141***d*, (v) key pair generation algorithms **141***e*, and/or (vi) a key derivation function **141***f*. The first server private key **105***c* could be utilized at step **1101** by any of (i) generating a server digital signature **506** that is sent to an application server **171**, (ii) receiving a symmetric key **127** for the secure connection, where the symmetric key **127** is decrypted by a set of servers **105***n* using the first server private key **105***c*, (iii) input into a key derivation function **141***f* including using ECIES with the first server private key **105***c* to obtain a derived shared secret key **129***b*, and (iv) the first server private key **105***c* is used to process a first server public key **114**, and the first server public key **114** is used to establish the secure connection. Other possibilities exist as well for values or settings specified in a set of cryptographic parameters **126** without departing from the scope of the present invention.

**[0327]** At step **1102**, in an exemplary embodiment the set of servers **105***n* can receive a first message **208** that includes a module identity **110**. The first message **208** could include a server instruction **414**, a module encrypted data **403**, and/or a module digital signature **405**. In an exemplary embodiment, server **105** can receive other messages **208** and module public keys **111** both before and after steps **1101** and step **1102**, as well as other steps contemplated herein. In other words, the various messages and responses illustrated in FIG. **11** can comprise subsets of all messages and responses, such that the subsets comprise illustrated embodiments of the present invention. Note that step **1102** could take place before or after steps **1101** and **1103**. Although not illustrated in FIG. **11**, a set of servers **105***n* could also receive can receive over time a series of module public keys **111** associated with module **101** including the module identity **110**. Each member of the series of module public keys **111** could be received in a message **208** that could also include a module public key identity **111***a* in order to track the module public keys **111** in the series. A set of servers **105***n* could receive the series of module public keys **111** associated with module **101** including the module identity **110** using a step **1004** depicted and described in connection with FIG. **10** above.

**[0328]** At step **1103**, a set of servers **105***n* can verify a module digital signature **405** with module identity **110** using a first module public key **111**. The first module public key **111** could be received and recorded by a set of servers **105***n* before or after step **1101**, including receiving the first module public key **111** with a module identity **110** from module **101** in a message **208**. Note that the module digital signature **405** does not need to be received in the message **208** received at step **1102**, and module digital signature **405** could be received in a different message **208**. In an exemplary embodiment, the common feature of steps **1102** and steps **1103** can comprise that a set of servers **105***n* performs the action, and a module **101** with a module identity **110** submitted the data illustrated in order for a set of servers **105***n* to perform the actions described in steps **1102** and **1103**.

**[0329]** At step **1104**, in an exemplary embodiment the set of servers **105***n* can send a first response **209** that includes server digital signature **506**, where server digital signature **506** is processed using a second server private key **105***c*, and the second server private key **105***c* can be different than the first server private key **105***c* used in step **1101**. As exemplary embodiments, (i) the first server private key **105***c* from a step **1101** could be an RSA-based key such as, but not limited to, a private key associated with an exemplary RSA-based public key depicted and described in connection with FIG. **1***g* of U.S. patent application Ser. No. 14/039,401, filed Sep. 27, 2013 in the name of John Nix, and (ii) the second server private key **105***c* from a step **1104** could be an ECC-based key such as, but not limited to, a private key associated with an exemplary ECC-based public key depicted and described in connection with FIG. **1***h* of U.S. patent application Ser. No. 14/039,401, filed Sep. 27, 2013 in the name of John Nix. Note that the second server private key **105***c* can also be associated with a second set of cryptographic parameters **126** that are different or not equal to a first set of cryptographic parameters **126** that are associated with the first server private key **105***c* used in a step **1101**. The second set of cryptographic parameters **126** could be used by a key pair generation algorithms **141***e* to process or derive the second server private key **105***c*. Also note that both the first server private key **105***c* used in step **1101** and the second server private key **105***c* used in step **1104** can each be associated with a different random number **128***a*, where the different random numbers **128***a* could also each be used by a key pair generation algorithms **141***e* to process or derive the first server private key **105***c* and the second server private key **105***c*, respectively.

**[0330]** At step **1105**, in exemplary embodiments the set of servers **105***n* can receive a second message **208** that includes a module identity **110**. The message **208** could include a server instruction **414**, a module encrypted data **403**, and/or a module digital signature **405**. At step **1106**, the set of servers **105***n* can send a second response **209** with a set of cryptographic parameters **126**, where module **101** can use the set of cryptographic parameters **126** to derive a second module public key **111** and a corresponding module private key **112**, potentially by using a step **515**. According to an exemplary embodiment, the set of cryptographic parameters **126** sent by a set of servers **105***n* in a step **1106** could be included in a server encrypted data **504**. Security of a system **100** and other systems herein can be increased by encrypting a set of cryptographic parameters **126** sent to a module **101**. In an exemplary embodiment, the set of cryptographic parameters **126** sent in a step **1106** can include at least one of (i) the name or value for a symmetric ciphering algorithm **141***b*, (ii) parameters or values for a module random seed file **139**, (iii) the name or value for an asymmetric ciphering algorithm **141***a*, (iv) the name or value for a digital signature algorithm **141***d*, (iv) a value for a key pair generation algorithm, (v) a name or value for an elliptic curve defining equation, and/or (vi) a value for a key derivation function **141***f*. Module **101** could use the set of cryptographic parameters **126** sent in a step **1106** with a key pair generation algorithms **141***e* and a random number generator **128** to derive the second module pub-

lic key **111**. Module **101** could use a step **515** to derive the second module public key **111** and a corresponding module private key **112**.

**[0331]** At step **1107**, in an exemplary embodiment a set of servers **105***n* can receive (i) the second module public key **111** and a module identity **110**, and (ii) verify the second module public key **111** using the first module public key **111** received in a step **1103**. In an exemplary embodiment, the a set of servers **105***n* can use the first module public key **111** to verify the received second module public key **111** using at least one of several sub-steps. The sub-steps at step **1107** to verify the second module public key **111** using the first module public key **111** could comprise any of (i) receiving the second module public key **111** and a module identity **110** with a module encrypted data **403** that uses a symmetric ciphering algorithm **141***b*, where the symmetric key **127** for encrypting and decrypting the module encrypted data **403** at step **1107** could previously be communicated before step **1107** using the first module public key **111** (such as a server **105** in the set of servers **105***n* sending the symmetric key **127** to module **101** in a server encrypted data **504**, where the server encrypted data **504** was ciphered with an asymmetric ciphering algorithm **141***a* and the first module public key **111**), (ii) receiving the second module public key **111** and module identity **110** with a module digital signature **405** where the module digital signature **405** is verified by the set of servers **105***n* using the first module public key **111** (and module **101** could process the module digital signature **405** with the module private key **112** for the first module public key **111** used in a step **1103**), and/or (iii) using a derived shared secret key **129***b* with a message digest authentication for verifying a received message **208** with the second module public key **111** at step **1107**, where the derived shared secret key **129***b* was processed using a key derivation function **141***f* and the first module public key **111**. Other possibilities exist as well without departing from the scope of the present invention for using the first module public key **111** from a step **1103** to verify the second module public key **111** at a step **1107**.

**[0332]** At step **1108**, in exemplary embodiments a set of servers **105***n* can decrypt a module encrypted data **403** using the verified second module public key **111**, where the second module public key **111** was verified in a previous step **1107**. The module encrypted data **403** be received in a message **208** and could include a server instruction **414**, sensor data **604***b*, a security token **410**, a timestamp **604***a*, and/or other data. The set of servers **105***n* can decrypt the module encrypted data **403** in a received message **208** at step **1108** using the second module public key **111**. In one embodiment, the module encrypted data **403** in step **1108** could be ciphered with a symmetric key **127**, where the symmetric key **127** was received in a prior module encrypted data **403** before step **1108** and the symmetric key **127** in the prior module encrypted data **403** before step **1108** could be (i) ciphered using an asymmetric ciphering algorithm **141***a* and the second module public key **111**, or (ii) ciphered using a symmetric ciphering algorithm **141***b* and a derived shared secret key **129***b*, where the derived shared secret key **129***b* was derived using the second module public key **111** and a key derivation function **141***f*. The symmetric key **127** received in a prior module encrypted data **403** before step **1108** with a module identity **110** could be recorded in a shared module database **105***k*. A set of servers **105***n*, including one member of the set of servers **105***n*, could access the shared module database **105***k* in order to obtain or read the symmetric key **127**. In another embodiment, the prior module encrypted data **403** received prior to step **1108** with the symmetric key **127** could be ciphered with a different key that was communicated using the second module public key **111**. Other possibilities exist as well without departing from the scope of the present invention for a set of servers **105***n* to use the second module public key **111** to decrypt a module encrypted data **403** in a step **1108**.

**[0333]** At step **1109**, a set of servers **105***n* can send sensor data **604***b* to an application server **171** and/or application **171***i* using the first server private key **105***c*. The sensor data **604***b* could be received in a module encrypted data **403**, such as but not limited to sensor data **604***b* that could be received in a module encrypted data **403** at step **1108**. The sensor data **604***b* could be sent to application server **171** and/or application **171***i* using a secure connection data transfer **802**, where the secure connection data transfer **802** was established via a secure connection data setup **801**, and the secure connection data setup **801** could use the first server private key **105***c* at step **1101**. A secure connection data transfer **802** using a first server private key **105***c* is depicted and described in connection with FIG. **8**. In this manner, (i) a set of servers **105***n* can use a first server private key **105***c*, with an associated set of cryptographic parameters **126**, to communicate with an application server **171** and/or application **171***i*, and (ii) a set of servers **105***n* can use a second server private key **105***c*, with a different associated set of cryptographic parameters **126**, to communicate with a module **101**. The first and second server private keys **105***c*, could each use a set of cryptographic parameters **126** that are selected in order to optimize or enhance a desired level of security and efficiency for communicating (i) with another server for the first server private key **105***c* and (ii) with a set of modules **101** for the second server private key **105***c*.

**[0334]** FIG. **12**

**[0335]** FIG. **12** is a simplified message flow diagram illustrating an exemplary system with exemplary data transferred between a module and an application using a server, in accordance with exemplary embodiments. System **1200** can include an application server **171**, a server **105**, and a module **101**. Although a single application server **171**, server **105**, and module **101** are illustrated in FIG. **12**, a system **1200** could include a plurality of any of these elements. Application server **171** can include application **171***i* and utilize IP:port **702** for communicating with server **105**. Although not illustrated in FIG. **12**, application server **171** could also communicate with other servers or nodes on the Internet, including a user **183** via a web portal **171***j* illustrated in FIG. **1***d*, or an enterprise resource planning (ERP) system (not shown). The nodes illustrated in FIG. **12** could communicate using Internet protocols such as, but not limited to, TCP and/or UDP, and the network between server **105** and application server **171** could be either via the public Internet **107** or a private intranet or a VPN layered on top of the public Internet **107**. Other possibilities exist as well, and according to an exemplary embodiment, application server **171** and server **105** can be connected via a LAN, such that packets between the two do not route over the public Internet **107**. Sending data between two nodes on a LAN can also be considered using a secure connection data transfer **802**.

**[0336]** Server **105** can include a module controller **105***x*, a shared secret key **510**, and a module identity **110**, in addition to the other components and values shown for a server **105** illustrated in FIG. **1***f* and FIG. **1***c*, including a module data-

base **105***k*, and one or more server private keys **105***c*. Shared secret key **510** is depicted and described in connection with FIG. **5***b*, and can also comprise a pre-shared secret key **129***a* in one embodiment where server **105** receives a module public key **111** from module **101** for the first time (i.e. before server **105** sends or receives encrypted data with module **101**). As contemplated herein, a server **105** may also comprise a set of servers **105***n*, such that the set of servers **105***n* can perform the actions depicted and described for a server **105** illustrated in FIG. **12** and other Figures. Module identity **110** can comprise an identity for module **101**, and is also depicted and described in connection with FIG. **1***e* and elsewhere herein. Server **105** can use IP:port number **901** for communicating with application server **171** and IP:port number **207** for communicating with module **101** and other modules. Note that server **105** could also use multiple IP:port numbers **901** and **207** in a system **1200**, such as, but not limited to, a first IP:port number **901** to communicate with a first application server **171** and/or application **171***i*, a second IP:port number **901** to communicate with a second application server **171**, a first IP:port number **207** to communicate with a first set of modules **101**, a second IP:port number **207** to communicate with a second set of modules **101**, etc. The IP addresses and port numbers within an IP:port number **901** and **207** can also change over time.

**[0337]** Module controller **105***x* is depicted and described in connection with FIG. **1***c*, FIG. **8**, FIG. **9**, and elsewhere herein. Module controller **105***x* can transmit, send, and receive packets for server **105** using IP:port number **207**. Although a single module controller **105***x* and server **105** are illustrated in FIG. **12**, a server **105** could include multiple module controllers **105***x* that are distributed, and server **105** could also be distributed such that different sub-servers **105***w* perform the function of server **105** in an exemplary embodiment, and the sub-servers **105***w* could also include the module controller **105***x*. Module **101** in system **1200** can comprise a module **101** as depicted and described in connection with FIG. **1***a*, FIG. **1***b*, FIG. **1** *d*, FIG. **2**, and elsewhere herein. Module **101** can include a module identity **110** and a shared secret key **510** and utilize an IP:port number **204** for sending and receiving data with server **105**. IP:port number **204** can also change over time, such that module **101** uses either a different IP address **202** or port number **203** when (i) sending one message **208** or a series of message **208** to (ii) sending the next message **208** or series of messages **208**. According to an exemplary embodiment, module **101** can connect with multiple different networks **102** over time and each network **102** may provide a different IP address **202** to module **101**. Alternatively, the same network **102** may provide a different IP address **202** for module **101** at different times.

**[0338]** In an exemplary embodiment, module **101** can use a different IP address **202** between either periods of sleep or when a DHCP lease expires, and other possibilities exist as well. As in other Figures in the present invention, IP addresses illustrated in FIG. **12** may comprise either IPv4 or IPv6 addresses. System **1200** may include a firewall **104**, which may operate between module **101** and server **105**, and firewall **104** can provide NAT routing functionality, such that IP address **210** illustrated in FIG. **12** is different than IP address **202** (in FIG. **2**), and firewall **104** may translate ports as well. Note that firewall **104** may also be a symmetric firewall **104** illustrated in FIG. **9**, such that addresses and ports are not translated by firewall **104**, but in this case IP:port **204** may change over time in exemplary embodiments (such as, but not limited to, from module **101** using different networks **102**, or acquiring different IP addresses **202** between periods of sleep, etc.). The IP address **210** and port number **605** illustrated in FIG. **12** for firewall **12** can comprise the IP address and port number on the external interface of firewall **104**, such that packets routed to/from the Internet **107** with module **101** could use IP address **202** and port number **605** as a source/destination IP:port number, respectively.

**[0339]** Prior to step **1201**, module **101** may optionally derive a module public key **111** and a module private key **112** using a step **515**, as depicted and described in connection with FIG. **5***b*. Note that the internal derivation of module public key **111** and module private key **112** using a step **515** are not required to use the other components and steps illustrated in a system **1200**. A system **1200** (or system **1300** below in FIG. **13**) can also be used in an alternative embodiment where module private key **111** is obtained by other means than internal derivation using a key pair generation algorithm **141***e*, such as loading module private key **112** into a nonvolatile memory **101***c* or **101***w* upon manufacturing, distribution, or installation or a module **101** or at other times. However, in an exemplary embodiment, a system **1200** illustrated in FIG. **12** can be useful for secure and efficient communication between a module **101**, server **105**, and application server **171** when module **101** also derives the module private key **112** and module public key **111**, potentially by using a step **515**. The derivation of keys does not need to use and/or be associated with IP:port **204**, and step **515** is illustrated in FIGS. **12** and **13** are shown for an exemplary sequence of timing and location of message flows, such that key derivation using a step **515** can take place before a step **1201**.

**[0340]** At step **1201**, in an exemplary embodiment server **105** can use a module controller **105***x* to receive a first message **208** that includes module public key **111**. The first message **208** can also include a module identity **110**, or other identifying information such that server **105** can determine the first message **208** with module public key **111** is associated with module identity **110**. The first message **208** can also preferably include a module public key identity **111***a* associated with the module public key **111**. Although not illustrated in FIG. **12**, at step **1201** module controller **105***x* can also receive a set of cryptographic parameters **126** associated with module public key **111**, such as, but not limited to, a value for an elliptic curve defining equation, a RSA modulus, a time-to-live value or expiration date, etc. As received by server **105** (i.e. after traversing firewall **104**), the source IP address and source port number in message **208** can comprise IP address **210** and port number **605**, which can be different than IP:port number **204** due to network address translation by a firewall **104**. Module controller **105***x* can use IP:port number **207** to receive the first message **208**, wherein IP:port number **207** can comprise a destination address in a packet header of the first message **208** as illustrated in FIG. **6***a*. Although depicted in FIG. **12** as a "first message" **208**, module **101** may have previously sent messages **208** to server **105**, and the "first message" **208** can comprise the first message **208** received within the series or sequence of packets illustrated in FIG. **12**. Other messages **208** may potentially flow before and/or after a "first message" **208**. This terminology of "first message", "second response", "second public key", etc. contemplated in various Figures herein may refer to the "first message", "second response", "second public key", "first set of parameters", etc. described in the illustrated flows within each Figure. Other messages, responses, keys, and parameters may be

communicated before and/or after a depicted "first message", "second response", "second public key", etc. and the depicted elements can comprise a subset of other messages, responses, keys, etc. that may also flow in addition to the elements illustrated.

**[0341]** Although server **105** is illustrated as receiving module public key **111** in FIG. **12** using a module controller **105***x*, a first sub-server **105***w* could receive module public key **111**, and a different sub-server **105***w* or server **105** using a different module controller **105***x* could send and/or receive subsequent messages and responses illustrated in FIG. **12**. Thus, a system **1200** could use a plurality of module controllers **105***x* in a coordinated manner to operate as a single module controller **105***x* illustrated in FIG. **12** and FIG. **13**. In an exemplary embodiment, server **105** can receive module public key **111** at a step **1201** due to any of (i) module **101** communicating with server **105** for the first time, (ii) module **101** deriving a new module public key such as using step **515** in FIG. **5***b*, or (iii) an end user, technician, or distributor loading a new module public key **111** (with a module private key **112**) into module **101**, and other possibilities for reasons for receiving module public key **111** exist as well without departing from the scope of the present invention. In an exemplary embodiment, server **105** has already securely communicated with module **101** using a different or prior module public key **111** (not shown, and also possibly with a different set of parameters **126**) before receiving the module public key **111** illustrated in FIG. **12**, and the module public key **111** at step **1201** can represent a new module public key **111** to be used with subsequent communications. In an exemplary embodiment, the module public key **111** received at step **1201** in FIG. **12** comprises an exemplary message **208** illustrated in FIG. **6***b*. Server **105** could also receive the new module public key **111** in FIG. **12** by previously sending a module instruction **502** for module **101** to derive a new module public key **111** and module private key **112** using a set of parameters **126**, and other possibilities exist as well.

**[0342]** At step **1202** in exemplary embodiments module controller **105***x* and/or server **105** can verify or authenticate module public key **111**, where the received data that includes module public key **111** also includes a received module identity **110**. Module controller **105***x* and/or server **105** could authenticate and/or verify module public key **111** is associated with the recorded module identity **110** using a step **517** depicted and described in connection with FIG. **5***b*, including using a shared secret key **510**. At step **1202**, module controller **105***x* can also use a set of cryptographic parameters **126** and a set of cryptographic algorithms **141** to verify or authenticate module public key **111** at step **1202**. In one exemplary embodiment, step **1202** could use a message digest authentication and the shared secret key **510** to verify a response from module **101** with the message digest. At step **1202**, module controller **105***x* and/or server **105** can take other actions besides message digest using the shared secret key **510** to determine if module **101** has the shared secret key **510**. Upon determination that module **101** has the shared secret key **510** (i.e. determining the received module public key **111** is associated with the recorded module identity **110** via the shared secret key **510**) then module public key **111** with the received module identity **110** can be authenticated and/or verified. In one exemplary embodiment module controller **105***x* and/or server **105** could use shared secret key **510** in processing a symmetric key **127**, such that if server **105** can decrypt data sent with module public key **111** and a symmetric ciphering algorithm **141***b*, then server **105** can determine that module **101** has the shared secret key **510**. Shared secret key **510** could also be used by module **101** in sending a secure hash signature with the module public key **111** at step **1201** (where shared secret key **510** is used by module **101** to generate the secure hash signature), and server **105** could verify the received secure hash signature with the shared secret key **510** and a secure hash algorithms **141***c* at step **1202**.

**[0343]** Other possibilities exist as well for authenticating and/or verifying module public key **111** at step **1202**, and the use of a shared secret key **510** is not required in order to authenticate and/or verify that module public key **111** is associated with a recorded module identity **110** at a step **1202**. A set of cryptographic parameters **126** that were received with module **101** in step **1201** could also specify the actions or processes that module controller **105***x* and/or server **105** can use to authenticate and/or verify module public key **111** at step **1202**. In an exemplary embodiment, server **105** can authenticate and/or verify module public key **111** is associated with module identity **110** using a certificate **122** and a signature from a certificate authority **118**, such as using a step **412** depicted and described in connection with FIG. **4**. In this embodiment with a certificate authority **118** or another server performing steps **1201** and **1202**, the certificate authority **118** or another server may operate in conjunction with server **105** and/or module controller **105***x* and perform an authentication and/or verification of module public key **111**. In another exemplary embodiment, server **105** and/or M2M service provider **108** (with potentially a different server than the server **105** illustrated in FIG. **12**) may have communicated with module **101** prior to receiving module public key **111** at step **1201**, and in this case server **105** could use a different key than shared secret key **510** to authenticate and/or verify at step **1201** module public key **111** illustrated in FIG. **12** is properly associated with module identity **110**, such as using the different key (not shown) with a message digest, module digital signature **405**, symmetric ciphering algorithm **141***b*, and/or secure hash algorithms **141***c* using data received with module public key **111**. A set of parameters **126** received with module public key **111** at step **1201** could specify that module controller **105***x* use the different key to authenticate and/or verify module public key **111** at step **1202**. Servers for a certificate authority **118**, M2M service provider **108**, module provider **109**, and/or server **105** connected via a network can also operate or function as a set of servers **105***n*.

**[0344]** In another embodiment, server **105** and/or M2M service provider **108** may previously have communicated a symmetric key **127** with module **101**, and the symmetric key **127** could be used to authenticate and/or verify module public key **111** at step **1202**. Server **105** could receive the symmetric key **127** from (i) the M2M service provider **108**, or (ii) module **101** before step **1201** (in a previous state where module **101** was authenticated with server **105**). Module public key **111** at step **1201** and/or other data could be sent in a module encrypted data **403** using the symmetric key **127**, and decrypting the module public key **111** from a step **1201** with the symmetric key **127** can determine that module public key **111** is authenticated and/or verified at a step **1202**.

**[0345]** In another embodiment, server **105** could have received a prior module public key **111** (possibly from M2M service provider **108** or another authenticated and/or verified source) before the received module public key **111** from step **1201** illustrated in FIG. **12**, and server **105** and/or module controller **105***x* could use the prior module public key **111** and

a module digital signature **405** (processing using a prior module private key **112**) sent with the module public key **111** at step **1201** illustrated in FIG. **12** to determine that the module public key **111** is authenticated and/or verified at a step **1202**. In accordance with exemplary embodiments, step **1202** can authenticate and/or verify that received module public key **111** is properly associated with the module identity **110** recorded in server **105** in order to securely communicate with module **101** at subsequent steps illustrated in FIG. **12**, thereby securing a system **1200**. Without proper authentication and/or verification that module public key **111** is properly associated with a recorded module identity **110** at step **1202** by using the exemplary embodiments described herein, a system **1200** may be vulnerable to an imposter, hackers, or fraudulent submissions of module public key **111** and/or subsequent communication with the wrong (or fraudulent) module **101** in subsequent communications.

**[0346]** At step **1202***a*, an application interface **105***i* can send the received and verified module public key **111** to application server **171** and/or application server **171***i* via a secure connection data transfer **802** and an application message **701**. The application message **701** can also include the module identity **110**, a module public key identity **111***a*, and a set of cryptographic parameters **126**, and the module identity **110**, the module public key identity **111***a*, and the set of cryptographic parameters **126** could be also received in the message **208** at step **1201**. Although not illustrated in FIG. **12**, application server **171** and/or application **171***i* could (i) record the data received in step **1202***a* and also (ii) send values received in application message **701** at step **1202***a* to a second server **105**, such as a second server **105** (not shown in FIG. **12**) illustrated in FIG. **1***h* using a second secure connection data transfer **802**. In this manner, a second server **105** can record the module identity **110**, the verified module public key **111**, the module public key identity **111***a*, and the set of cryptographic parameters **126** associated with module public key **111** in a module database **105***k* associated with the second server **105**.

**[0347]** In another embodiment, at step **1202***a*, server **105** could record the module identity **110**, the verified module public key **111**, the module public key identity **111***a*, and the set of cryptographic parameters **126** associated with module public key **111** in a shared module database **105***k* (such as the module database **105***k* illustrated in FIG. **1***h*), and a second server **105** could have access to the data by querying the shared module database **105***k*. A shared module database **105***k* is also illustrated in FIG. **1***h*. In an embodiment where server **105** accesses a shared module database **105***k*, at step **1202***a* the module public key **111** and related data (such as, but not limited to, module identity **110**), could be sent to the shared module database **105***k* via the message shown in step **1202***a* instead of sending the message to application server **171**. In this case by using a shared module database **105***k*, the message at step **1202***a* could comprise or trigger an "insert" command or message to shared module database **105***k*, where the insert command could include (i) a table name, (ii) the received and verified module public key **111**, (iii) the module identity **110**, (iv) a set of cryptographic parameters for module **101**, and (v) a module public key identity **111***a*. In this manner with exemplary embodiments, by sharing module database **105***k* with multiple servers **105**, communication with module **101** and a plurality of servers **105** can be more efficient, since each server **105** can access data such as a previously recorded and verified module public key **111**, module identity **110**, module public key identity **111***a*, and set of cryptographic parameters **126** associated with each module **101**. By reducing the transmission of messages between a module **101** and different servers **105** over the lifetime of module **101**, a system can be simultaneously more secure and more efficient.

**[0348]** By a second server **105** receiving the values from application server **171** and/or application **171***i*, the second server **105** can also record that module **101** with the module public key **111** (received in step **1201** shown in FIG. **12**) and module identity **110** has been verified. In addition, although server **105** is illustrated as sending the module public key **111** to application server **171** in FIG. **12** at step **1202***a*, server **105** could alternatively send the module public key **111**, the module identity **110**, a module public key identity **111***a*, and a set of cryptographic parameters **126** associated with module **101** directly to a second server **105**, such as, but not limited to, server B **105** depicted and described in connection with FIG. **1***h*. Server **105** could send the module public key **111**, the module identity **110**, a module public key identity **111***a*, and a set of cryptographic parameters **126** associated with module **101** to the second server B **105** illustrated in FIG. **1***h* using a secure connection data transfer **802**. In another embodiment, a second server B **105** illustrated in FIG. **1***h* could access the verified module public key **111** from steps **1201** and **1202** by accessing a module database **105***k* where server **105** had recorded the module public key **111**, the module identity **110**, a module public key identity **111***a*, and a set of cryptographic parameters **126** associated with module **101**.

**[0349]** At step **1203**, module controller **105***x* can send a server digital signature **506** with a server identity **206**, and server digital signature **506** could be processed as described in FIG. **5***a*. In an exemplary embodiment, module controller **105***x* and/or server **105** could use (i) a first server private key **105***c*, (ii) a set of cryptographic parameters **126**, and (iii) a digital signature algorithm **141***d* to process and/or create server digital signature **506**. Module controller **105***x* can use IP:port number **207** for sending server digital signature **506**. In an exemplary embodiment, a second server private key **105***c* using a different set of cryptographic parameters **126** could be used with secure connection setup **801** to application **171***i*. In one embodiment, first server private key **105***c* can use ECC algorithms **154** and the second server private key **105***c* can use RSA algorithms **153**, and the selection of ECC algorithms **154** or RSA algorithms **153** can be specified in a set of cryptographic parameters **126**, although other possibilities exist as well. In another embodiment, a single server private key **105***c* (possibly with a single set of cryptographic parameters **126**) can be used for both server digital signature **506** and secure connection setup **801**. Server digital signature **506** could be sent within a response **209**, and although not illustrated in FIG. **12**, server digital signature **506** could be associated with or sent with a packet that includes a symmetric key **127** for use by module **101**. A set of parameters **126** used with processing server digital signature **506** can be sent by module controller **105***x* before, with, or after step **1203**.

**[0350]** Although not illustrated in FIG. **12**, module controller **105***x* could also send server encrypted data **504** to module **101** at step **1203**, where the server encrypted data **504** can include exemplary values such as, but not limited to, an actuator instruction **706**, a module instruction **502**, a security token **401**, additional cryptographic parameters **126**, a symmetric key **127**, or an acknowledgement that a message **208** has been received. In accordance with a preferred exemplary embodi-

ment, module controller **105***x* sends a server digital signature **506** with server identity **206** to module **101** in a step **1203** upon the successful authentication and/or verification of module public key **111** received in step **1202** above. Module **101** can use the server digital signature **506**, server public key **114**, a set of parameters **126**, and a digital signature algorithm **141***d* to verify the identity of server **105** used in FIG. **12**. Server digital signature **506** may also be sent with or as a signal that the module public key **111** was properly received in a step **1201** and/or authenticated or verified at a step **1202**. In one embodiment, the set of parameters **126** can specify (i) a secure hash algorithm **141***c* to use with a digital signature algorithm **141***d* (such as, but not limited to, either SHA-256, SHA-3, etc.), and (ii) additional details for processing a secure hash algorithm **141***c* used with a digital signature algorithm **141***d* such as the format or order of strings or values input into a secure hash algorithm **141***c*. In another embodiment, digital signature algorithms **141***d* can include logic for the format, order, strings or values, and encoding to use for processing a digital signature including a server digital signature **506** and/or a module digital signature **405**. In other words, a set of parameters **126** and/or digital signature algorithms **141***d* can include settings such that a server **105** and module **101** can calculate, derive, or process the same digital signature as the other node.

**[0351]** At step **1204**, a module controller **105***x* can receive a module digital signature **405**. Module controller **105***x* can use an IP:port number **207** to receive the module digital signature **405**. Although not illustrated in FIG. **12**, module controller **105***x* can receive module digital signature **405** with a module identity **110**. In addition, although the receipt of module digital signature **405** is depicted as after the sending of server digital signature **506**, module digital signature **405** could be received before the sending of server digital signature **506** in exemplary embodiments. Processing a module digital signature **405** is also depicted and described in connection with FIG. **4** above and elsewhere herein. Module digital signature **405** could be received in a message **208**, which could be formatted with the exemplary format for a message **208** in illustrated in FIG. **6***a*, and other possibilities exist as well.

**[0352]** Module digital signature **405** received in step **1204** can also include a symmetric key **127**, and symmetric key **127** could be ciphered using an asymmetric ciphering algorithm **141***a*, where a module **101** used a server public key **114** in order to encrypt the symmetric key **127**. Symmetric key **127** could be used with a symmetric ciphering algorithm **141***b* and a set of parameters **126** at subsequent (i) step **1207** to decrypt data within a second message **208** and/or (ii) step **1208** to encrypt data within a response **209**. The use of a symmetric key **127** with a set of parameters **126** is depicted and described in connection with FIG. **1***e*, FIG. **3**, FIG. **4**, and FIG. **5***a*, and elsewhere herein. In an exemplary embodiment, the receipt of module digital signature **405** at step **1204** can be omitted, and the verification of messages or packets from module **101** can be processed with other means, such as using the shared secret key **510**. However, over time and with a change of the use of IP addresses such as, but not limited to, changing source IP:port number **210**:**605**, and possibly using different sub-servers to receive messages from module **101**, the periodic transmission of a module digital signature **405** may be preferred.

**[0353]** At step **1205**, module controller **105***x* can verify the module digital signature **405** for module identity **110** received in step **1204** using the module public key **111** (*i*) received in step **1201** and (ii) authenticated in step **1202**. At step **1205**, module controller **105***x* can use a set of parameters **126**, the module public key **111** received in step **1201**, digital signature algorithms **141***d*, secure hash algorithms **141***c*, and module identity **110** in order to verify module digital signature **405**. Although not illustrated in FIG. **12**, after step **1205**, module controller **105***x* can then follow the sequence of steps illustrated in FIG. **3** to process additional messages **208** and responses **209** from/to module **101**, until application message **701** with module instruction **502** is received at step **1206** below. The symmetric key **127** received in step **1204** with a module digital signature **405** verified in a step **1205** could be used to receive module encrypted data **403** and send server encrypted data **504** after step **1205**.

**[0354]** At step **1205***a*, after verifying the module digital signature **405** received in step **1204** and verified in step **1205**, an application interface **105***i* can send a symmetric key **127** for use with module **101** to application server **171** and/or application server **171***i* via a secure connection data transfer **802** and an application message **701**. The symmetric key **127** sent in application message **701** could be received in step **1204**, or the symmetric key **127** could be processed or generated by server **105** and sent to module **101** in step **1203**. The application message **701** at step **1205***a* can also include the module identity **110**. Although not illustrated in FIG. **12**, application server **171** and/or application **171***i* could record the data received in step **1205***a* and also send values received in application message **701** at step **1205***a* to a second server **105**, such as a second server **105** (not shown in FIG. **12**) illustrated in FIG. **1***h* using a second secure connection data transfer **802**. In this manner, a second server **105** can record include the module identity **110** and the symmetric key **127** with a module database **105***k* for the second server **105**. In another embodiment, at step **1205***a*, server **105** could record the received symmetric key **127** and module identity **110**, plus optionally related additional information, in a shared module database **105***k*. A second server **105**, such as server B **105** illustrated in FIG. **1***h*, could access the symmetric key **127** and module identity **110** via the shared module database **105***k*.

**[0355]** By a second server **105** receiving the symmetric key **127** and module identity **110** from either (i) application server **171** or (ii) a shared module database **105***k*, the second server **105** can communicate with module **101** using module identity **110** and the symmetric key **127** without the second server previously conducting the steps **1201** through **1204**. In this manner according to a preferred exemplary embodiment, a system **100** can be made more efficient, since a second server **105** (such as the second server **105** illustrated as "Server B" in FIG. **1***h*) can use the symmetric key **127** received from application server **171** or shared module database **105***k* to communicate with module **101** without requiring sending or receiving additional packets in order to establish or communicate a symmetric key **127**. In addition, although server **105** is illustrated as sending the symmetric key **127** to application server **171** in FIG. **12**, server **105** could alternatively send the symmetric key **127** directly to a second server **105**, such as server B **105** depicted and described in connection with FIG. **1***h*. Server **105** could send the symmetric key **127** and module identity **110** to the second server B **105** illustrated in FIG. **1***h* using a secure connection data transfer **802**. Server **105** could use a step **1205***a* to send the symmetric key **127** and module identity **110** to a plurality of servers, including and of a

plurality of application servers **171**, other servers **105**, and a one or more shared module databases **105***k*.

**[0356]** At step **1206**, in an exemplary embodiment server **105** can receive an application message **701** that includes a module instruction **502**. In exemplary embodiments, application message **701** received also includes a module identity **110** in order to specify which module **101** from a plurality of modules **101** that module instruction **502** is intended as the ultimate recipient. The application message **701** could be received through secure connection data transfer **802**, which could be established using secure connection setup **801**. Application interface **105***x* can use IP:port number **901** to receive application message **701**, where application message **701** includes IP:port number **702** as a source IP:port number in the packet header of application message **701**, as illustrated in FIG. **7**. Secure connection data transfer **802** and secure connection setup **801** are depicted and described in connection with FIG. **8** above. As contemplated herein, the term "secure connection" can refer to either secure connection data transfer **802** or secure connection setup **801**. An application message **701** is depicted and described in connection with FIG. **7** through FIG. **9**. A module instruction **502** within application message **701** can include an instruction, command, or data for module **101**, possibly including any of the exemplary module instructions **502** depicted and described in connection with FIG. **7** for an exemplary module instruction **502** in an application message **701**. As illustrated in FIG. **12**, application message **701** received in step **1206** can be received after a first message **208** and before a second message **208**.

**[0357]** In the exemplary embodiment of system **199** illustrated in FIG. **1***h*, the use of a server **105** for routing module instruction between (i) an application **171***i* and/or application server **171** and (ii) a module **101** may be preferred for many different reasons, including supporting scalability, increasing security, reducing bandwidth, supporting legacy cryptographic algorithms **141** supported on application server **171**, and/or increasing efficiency by using two different cryptographic schemes, where one is optimized for communication between modules and servers, and a second is optimized for communication between servers.

**[0358]** In exemplary embodiments, application message **701** from step **1206** can be received when (i) module **101** comprises a sleep or dormant state, (ii) a firewall port binding timeout value **117** associated with firewall **104** has expired, and/or (iii) communication with module **101** is not available for other reasons (such as, but not limited to, out of range of a wireless network, waiting for a battery **105***k* to be recharged, etc.) For any of the above cases, outbound packets sent from module controller **105***x* would not normally be received by module **101**. Consequently, after receiving application message **701** in step **1206**, module controller **105***x* can begin waiting for a wait interval **703**. As illustrated in FIG. **12**, application interface **105***i* can send application **171***i* a second application message **701** upon waiting for a second message **208** from module **101**, where the second application message **701** can comprise an application update **704** instruction of "waiting". Application update **704** instruction of "waiting" can include module identity **110**. In this manner, application **171***i* can be informed that module controller **105***x* and/or server **105** comprise a state of wait interval **703** for the next or second message **208** from module **101**. Wait interval **703** can end when the second message **208** is received from module **101** where the second message **208** can include a module identity **110** received in the application message **701** at step **1206**. The second message **208** can comprise the next message **208** after server **105** receives the module instruction **502**. Module controller **105***x* and/or server **105** can stop waiting when the next message **208** is received from module **101**, and the next message **208** is illustrated as a second message **208** in FIG. **12**.

**[0359]** At step **1207**, module controller **105***x* can receive a second message **208** from module **101**. Module controller **105***x* can receive the second message **208** by monitoring an IP:port number **207**. IP:port number **207** in step **1207** can be the same value or address as IP:port number **207** in step **1201**, or IP:port number **207** in step **1207** could be a different value or address than IP:port number **207** in step **1201**. According to exemplary embodiments, over time a specific address and/or numeric value for a port number used in an IP:port number contemplated herein can change. In an exemplary embodiment, the second message **208** includes a source IP address of IP address **210** and a source port number of port number **605**. As illustrated in FIG. **12**, IP address **210** and port number **605** can comprise an IP address and port number associated with the external interface of a firewall **104**, and can represent a source IP:port number in a packet header received in a message **208** as illustrated in FIG. **6***a* and FIG. **2**. In this exemplary embodiment, the numeric values for IP:port number **210**:**605** received in the second message **208** can be different than the numeric values for IP:port number **210**:**605** received in the first message **208**. IP:port number **210**:**605** received in the second message **208** can be different than in the first message due to many factors, including (i) module **101** has moved to a different network **102** in step **1207** after server **105** received the first message in step **1201**, (ii) firewall **104** has changed a source port number **605** based on logic internal to firewall **104** for allocating, sharing, using, and reusing port numbers with a plurality of connected nodes, and (iii) a network **102** could also change the IP address **202** used by module **101**.

**[0360]** Continuing at step **1207**, the second message **208** can preferably include a module identity **110**, wherein the module identity **110** was previously verified as being associated with module public key **111** in step **1202**. Module identity **110** in a second message **208** could comprise string or number with a different value than a module identity **110** received in the first message **208** at step **1201**, such as, but not limited to, the module identity **110** in the second message comprising a session identifier associated with module identity **110**. In exemplary embodiments, module controller **105***x* can process the string or number for module identity **110** received in the second message **208** in order to associate the string or value in a module identity **110** received in the second message **208** at step **1207** with the string or value for a module identity **110** received in the first message **208** at step **1201**. As contemplated herein, exemplary embodiments contemplate the use of different strings or values for the same module identity **110**. Different strings or values for a first module identity **110** can be separated from different strings or values for a second module identity **110** because (i) the strings or values as a first module identity **110** can be associated with a first physical module **101**, including possibly a serial number for the first physical module **101**, whereas (ii) the strings or values as a second module identity **110** can be associated with a second physical module **101**, including possibly a serial number for the second physical module **101**. Although not illustrated in FIG. **12**, the second message **208** at a step **1207**

could include additional data such as, but not limited to, a module encrypted data **403**, a module digital signature **405**, and/or a server instruction **414**. The server instruction **414** could be sent as plaintext in a body **602** or could be encrypted or obfuscated.

**[0361]** At step **1208**, in exemplary embodiments module controller **105***x* can send a response **209** to the second message **208**, and the response **209** can include the module instruction **502** received at step **1206**. The second message **208** and the response **209** can be sent and received as UDP packets or datagrams. In an exemplary embodiment, module controller **105***x* uses both (i) IP:port number **207** that received the second message **208** as a source IP:port number in response **209**, and (ii) the IP:port number **210**:**605** received in the second message **208** as a destination IP:port number in response **209**. In this manner, response **209** can traverse a firewall **104** in order to be received by module **101**. In an exemplary embodiment, module controller **105***x* can send response **209** before the expiration of a firewall port-binding timeout value **117**, where the start of firewall port-binding timeout value **117** began when message **208** traversed firewall **104**. Module instruction **502** in response **209** can be formatted or encoded differently than module instruction **502** received in application message **701** at step **1206**. Response **209** could include module instruction **502** within a server encrypted data **504**, where server encrypted data **504** can be ciphered using the symmetric key **127** received in step **1204**. In an exemplary embodiment, module instruction **502** can be sent as plaintext in response **209**, and in this case response **209** can preferably include a server digital signature **506** in order for module **101** to confirm or verify that server **105** and/or module controller **105***x* sent module instruction **502**.

**[0362]** According to an exemplary embodiment, at step **1209**, module controller **105***x* can receive a server instruction **414** comprising an acknowledgement with a timestamp **604***a* when module **101** properly received and/or executed module instruction **502** from step **1208**. Application interface **105***i* can send an application update **704** with the module identity **110**, where application update **704** can comprise (i) an acknowledgement that module **101** with module identity **110** executed the module instruction **502** received in step **1206**, and (ii) a timestamp value **604***a* when module **101** properly received and/or executed module instruction **502**. In exemplary embodiments, the inclusion of timestamp **604***a* can be important or useful for application **171***i* to manage or control a plurality of modules **101** via a server **105** or a set of servers **105***n*. The timestamp **604***a* can be useful because module **101** may utilize sleep and/or dormant states, or possibly having periodic outages or loss of access to Internet **107** and/or network **102**. As one example, there could also be an exemplary delay of minutes or longer between module **101**'s execution of module instruction **502** and when module **101** can send an acknowledgement such as server instruction **414**, possibly due to a sleep state or network outage. Additional unknown or uncertain time for application **171***i* between sending module instruction **502** at step **1206** the execution of module instruction **502** by module **101** can include the wait interval **703**. Consequently, in accordance with a preferred exemplary embodiment, server instruction **414** and application update **704** at step **1209** include a timestamp **604***a* that module **101** executed module instruction **502**.

**[0363]** Although not illustrated in FIG. **12**, server instruction **414** could include additional data such as sensor data **604***b* or other data associated with a state of module **101** or a component within module **101** including the state or value for an actuator **101***y*. Server instruction **414** could be included in a module encrypted data **403**. Application update **704** can include the sensor data **604***b* and additional information. In one embodiment, server instruction **414** can be received with a module identity **110** and server instruction **414** with timestamp **604***a* can be encrypted in a module encrypted data **403**.

**[0364]** FIG. **13**

**[0365]** FIG. **13** is a simplified message flow diagram illustrating an exemplary system with exemplary data transferred between a module and an application using a server, in accordance with exemplary embodiments. System **1300** can include an application server **171**, a server **105**, and a module **101**. Although a single application server **171**, server **105**, and module **101** are illustrated in FIG. **13**, a system **1300** could include a plurality of any of these elements. System **1300** illustrated in FIG. **13** can comprise the same components, steps, and message flows as system **1200** illustrated in FIG. **12**, with changes to support (i) a polling of a module instruction **502** from application **171***i* and/or application server **171** in a step **1302** instead of (ii) receiving module instruction **502** in a step **1206** illustrated in FIG. **12**.

**[0366]** Steps **1201** through step **1205** illustrated in FIG. **13** can comprise the same steps **1201** through **1205** depicted and described in connection with FIG. **12**. At step **1301**, application **171***i* can process a module instruction **502** to be sent to a module **101** with a module identity **110**. The determination and/or processing of a module instruction **502** could be for any reason application **171***i* prefers for a state, value, or setting within a module **101** to change. Module instruction **502** could comprise any of the exemplary module instructions **502** depicted and described in connection with FIG. **7**, and other possibilities exist as well. In one exemplary embodiment, at step **1301** application **171***i* could determine that a setting for an actuator **101***y* should change, such as based on the input from a user **183** or other automated control decisions. Although not illustrated in FIG. **12** above, an application **171***i* and/or application server **171** within a system **1200** could also use a step **1301** before sending the application message **701** with module instruction **502** in step **1206**, where the module instruction **502** can be processed at a step **1301** in a system **1200** above.

**[0367]** According to an exemplary embodiment, illustrated in FIG. **13**, after step **1301** the wait interval **703** illustrated in FIG. **12** can be omitted by not receiving module instruction **502** in an application message at step **1206**. Instead, and as illustrated in FIG. **13**, module controller **105***x* could receive the second message **208** from a step **1207** after receiving the module public key **111** in a step **1201**. Upon or after receiving the second message **208** in a step **1207**, server **105** and/or application interface **105***i* can send application **171***i* and/or application server **171** an application message **701** with a polling request at step **1302**. In an exemplary embodiment, the polling request could signal that a module **101** in a system **1300** is ready and available to receive the module instruction **502**. In an exemplary embodiment, application message **701** in a step **1302** can be sent using application interface **105***i* and an IP:port number **901**. The application message **701** can optionally be sent using a secure connection data transfer **802**. The polling request in application message **701** at step **1302** can be useful since module **101** may use periods of sleep or dormancy, and or periodically not be connected or accessible through a network **102** and/or firewall **104**, and in this case

module instruction **502** could not be transmitted or sent to module **101** at arbitrary times.

**[0368]** At step **1303**, application **171***i* can send the module instruction **502** processed above at step **1301**, after receiving the first application message **701** with the polling request. Module instruction **502** can be sent from application **171***i* to application interface **105***i* in a second application message **701** at step **1303** and may also use a secure connection data transfer **802**. Application interface **105***i* can receive the second application message **701** with the module instruction **502**. As illustrated in FIG. **13**, module controller **105***x* can then use a step **1208** to send the module instruction **502** in a response **209**. System **1300** can then also use step **1209** to receive a server instruction **414** with an acknowledgement and a timestamp **604***a*, and send an acknowledgement **704** to application **171***i* and/or application server **171**. In an exemplary embodiment, a step **1209** can be optionally omitted, and timestamp **604***a* could be optionally communicated in a separate application message **701** at a later time or within other application messages **701** not depicted in system **1300**.

**[0369]** System **1300** illustrated in FIG. **13** may be preferred in a system where a single application **171***i* communicates with a server **105**. In another embodiment, where multiple application servers **171** communicate with a server **105** or a set of servers **105***n*, then a system **1200** may be preferred, where server **105** receives the module instruction without sending an application message **701** with a polling request. One reason is that when multiple application servers **171** communicate with server **105**, an application interface **171***i* may not know the correct or proper application server **171** in order to send the application message **701** with the polling request at a step **1302** (since potentially many different application servers **171** may be a source of the module instruction **502**). Without knowing the proper application server **171** to send the application message **701** with the polling request, server **105** would then need to poll the plurality of application servers **171** each time a second message **208** was received in a step **1207**. A server **105** could receive many second messages **208** over time, and polling a plurality of application servers **171** each time a second message **208** was received could significantly increase network traffic and load, and therefore may not be efficient.

**[0370]** According to an exemplary embodiment, application **171***i* can operate within server **105**, and in this case IP:port **702** and/or IP:port **901** could be a loopback address and port number, which is reserved for the block of IPv4 addresses 127.x.x.x, and a similar loopback port for IPv6 addresses could be utilized as well when an application **171***i* operates within server **105**.

**[0371]** FIG. **14**

**[0372]** FIG. **14** is a graphical illustration of an exemplary system that includes a set of application servers, a set of servers, and a set of modules, in accordance with exemplary embodiments. System **1400** can include multiple application servers **171**, multiple servers **105**, and a plurality of modules **101**. A large, distributed system of thousands or more modules may utilize multiple application servers **171**, such as, but not limited to, the multiple application servers **171** associated with one or more enterprises with multiple operating divisions. Or, the application servers **171** in a system **1400** could each be associated with the same enterprise, while separate application servers **171** could be associated with separate divisions. In another embodiment, M2M service provider **108** may support different customers, where some customer may prefer or require the use of a different or segmented application server **171** and/or application **171***i*, and in this case an M2M service provider **108** with a plurality of modules **101** and servers **105** may utilize different application servers **171** for different customers. Multiple application servers **171** could comprise a set of application servers **171**. Other possibilities exist as well for using a set of application servers, a set of servers, and a set or plurality of modules without departing from the scope of the present invention.

**[0373]** In an embodiment where multiple application servers **171** communicate with multiple servers **105**, a combination of steps with system **1200** illustrated in FIG. **12** and steps within system **1300** illustrated in FIG. **13** may be preferred. Before step **1206** in FIG. **14** (illustrated as a "First" step in a system **1400**), any of the application servers **171** can process a module instruction **502** to be sent to a module **101** with a module identity **110**. With a plurality of application servers **171**, each application server can include or be associated with an application server identity **1401**. The determination and/or processing of a module instruction **502** could be for any reason application **171***i* and/or application server **171** prefers for a state, value, or setting within a module **101** to change. Note that any of the illustrated application servers **171** could originate the module instruction **502**, and thus a server A **105** may not know beforehand which of the multiple application servers to poll upon receipt of a message **208** from module **101** (which could comprise a second message **208** illustrated in at step **1207** in FIG. **12** and FIG. **13**). At step **1206** in FIG. **14**, the application server **171** processing or originating module instruction **502** can send the module instruction **502**, the module identity **110**, and the application server identity **1401** to a shared module database **105***k*. The module instruction **502** could be sent via a secure connection data transfer **802**, and other possibilities exist as well, including application server **171** having a remote connection to shared module database **105***k*. Application **171***i* could take an action (including an HTTP post or similar message) that would trigger an insertion of the module instruction **502** with module identity **110** into a database table within a shared module database **105***k*. Application **171***i* and/or application server **171** could also issue an insert command with module instruction **502**, such as with, but not limited to, an structured query logic (SQL) "insert" command.

**[0374]** The second step in a system **1400** could comprise module **101** sending a message **208** to a member of the set of servers. Message **208** could traverse a firewall **104** and be received by server A **105**. Message **208** can (i) include a module encrypted data **403** with a sensor measurement **604***a* and (ii) be sent after a module **101** changes from a sleep or dormant state to an active state. Message **208** could also be received by server A **105** after network **102** and/or Internet **107** connectivity was restored for module **101** after a period of network outage. A message **208** could be similar to the exemplary messages **208** illustrated in FIG. **2**, FIG. **6***a*, FIG. **6***b*, FIG. **9**, and other possibilities exist as well. Message **208** can include a module identity **110**. Message **208** in system **1400** could be a second message **208** illustrated at step **1207** of FIG. **12** and FIG. **13**. In an exemplary embodiment, message **208** in a system **1400** is sent as a UDP datagram **601***a* with forward error correction, such that module **101** sends multiple copies of the UDP datagram **601***a* (such as, but not limited to, sending an exemplary 3 copies of the same UDP datagram **601***a*) and server **105** can receive at least a subset of the multiple copies of the UDP datagram **601***a*. Although not

illustrated in FIG. **14**, server A **105** may preferably already have taken steps before receiving message **208** for (i) verifying a module public key **111** and module identity **110** for module **101** and/or (ii) authenticating communication with module **101** such using a symmetric key **127** to encrypt/decrypt data. Although a message **208** is illustrated in the second step in a system **1400**, server A **105** could receive other datagrams from a module **101** besides a message **208**.

**[0375]** The third step in a system **1400** can comprise server A **105** performing a step **1302** to poll shared module database **105***k* for an incoming module instruction **502** for module identity **110**. Note that server A **105** may have entered a waiting state or used a wait interval **703** (for communications related to module **101** with module identity **110**) before performing a step **1302** to poll shared module database **105***k*. In accordance with a preferred exemplary embodiment, server A **105** may wait until after receiving message **208** (illustrated as the "Second" step in FIG. **14**) before performing step **1302** because (i) before that time the module instruction **502** may not be sent to module **101** due to firewall **104**, and (ii) server A **105** may prefer to allow for time up until response **209** is sent for a shared module database **105***k* to collect all incoming module instructions **502** from all sources. Step **1302** can utilize a secure connection data transfer **802** or server A **105** may be able to perform SQL queries or similar commands directly with shared module database **105***k*. As contemplated herein and illustrated in FIG. **14**, a server **105** can use an application interface **105***x* to send and receive data with a shared module database **105***k*.

**[0376]** The fourth step in a system **1400** can comprise server A **105** receiving the module instruction **502** for module identity **110** from the shared module database **105***k* using a step **1303**. The module instruction **502** was received by shared module database **105***k* from application server **171** using a step **1206** illustrated above. Note that any of server A **105** and server B **105** could use a step **1302** and step **1303** to communicate with shared module database **105***k*, after receiving a message **208** or other data from a module **101**. In an exemplary embodiment, step **1303** may also comprise a server **105** receiving the application server identity **1401** with the module instruction **502** and module identity **110**. By acquiring the application server identity **1401** at a step **1303**, a server **105** can record the proper application server **171** to send an acknowledgement and a timestamp **604***b* at a subsequent time after successfully sending module instruction **502** to module **101**.

**[0377]** The fifth step in a system **1400** can comprise server A **105** sending a response **209** to the module **101** with module identity **110**, and the response **209** can include the module instruction **502**. The module instruction **502** could be included in a server encrypted data **504**. In an exemplary embodiment, response **209** is sent as a UDP datagram **601***b* with both (i) forward error correction and (ii) with a destination IP:port number in the UDP datagram **601***b* equal to a source IP:port number in a UDP datagram **601***a* received for the message **208**. In exemplary embodiments, response **209** is sent before the expiration of a firewall port-binding timeout value **117**.

**[0378]** The sixth step in a system **1400** can comprise server A **105** receiving a server instruction **414** of an acknowledgement and/or confirmation the module instruction **502** was properly executed or processed by module **101**, including a timestamp **604***a*. Server instruction **414** at the sixth step could be sent in a second message **208** that also includes module identity **110**, and server instruction **414** could also be in a module encrypted data **403**. Timestamp **604***a* could represent a time value associated with the processing of module instruction **502** by module **101**, such as, but not limited to, the time when module **101** implemented an actuator setting **706**, collected a sensor measurement **604***b*, and other possibilities exist as well. The timestamp **604***a* could be valuable for an application server **171** in order to keep track of the state of module **101** and/or a monitored unit **119**, since there can be delays between when application server **171** originated a module instruction **502** and when module **101** executed or applied module instruction **502** (in addition to delays when application server **171** can receive a confirmation the module instruction **502** has been executed).

**[0379]** The seventh step in a system **1400** can comprise server A **105** sending an application message **701** with the timestamp **604***a* to application server A **171**. Application message **701** can also include the module identity **110**. Note that server A **105** can obtain the proper application server **171** for sending application message **701** using the application server identity **1401** received by server A **105** in a step **1303** above. Application message **701** could also comprise an acknowledgement that module **101** properly executed the module instruction **502**.

## CONCLUSION

**[0380]** 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.

What is claimed is:

**1**. A method for supporting machine-to-machine communications, the method performed by a set of servers, the method comprising:

receiving a first message that includes a module identity and a first source Internet protocol address and port (IP:port) number, wherein the module identity is verified using at least one of a shared secret key and a module digital signature;

sending a response to the first source IP:port number, wherein the response includes a set of cryptographic parameters for deriving a public key and a private key;

receiving a series of different module public keys for the module identity;

receiving via a secure connection a module instruction and the module identity; and,

waiting until after (a) receiving a second message, wherein the second message includes a second source IP:port number and the module identity, before (b) sending the module instruction within a server encrypted data to the second source IP:port number, wherein the server encrypted data is ciphered using a module public key from the series.

**2**. The method of claim **1**, wherein the set of cryptographic parameters includes at least one of (i) an elliptic curve cryptography (ECC) named curve, (ii) a value for an ECC defining equation, (iii) a modulus for an RSA algorithm, (iv) a time-to-live value, (v) a value for a random number seed, (vi) a value for a module public key identity, (vii) a value for a module random seed file.

**3**. The method of claim **1**, further comprising encrypting the set of cryptographic parameters with a symmetric key, and sending the encrypted set of cryptographic parameters in the response.

**4**. The method of claim **1**, wherein the set of servers includes at least a first server and a second server, wherein the first server receives the first message and the second server receives the second message, wherein the set of servers includes a shared module database, wherein the shared module database receives the module instruction, and wherein the second server queries the shared module database for the module instruction after receiving the second message.

**5**. The method of claim **1**, wherein a module with the module identity sends (i) the first and the second messages, and (ii) the series of different module public keys, and wherein the module derives the series of different module public keys using at least a set of cryptographic algorithms and the set of cryptographic parameters.

**6**. The method of claim **6**, wherein the module comprises at least one of a mobile phone and a payment terminal, wherein one of the mobile phone and the payment terminal includes a universal integrated circuit card (UICC), wherein the UICC records an initial module private key, and wherein the set of servers records the initial module public key and uses the initial module public key to verify the module identity.

**7**. The method of claim **1**, wherein the series of different module public keys comprises at least a first module public key and a second module public key, wherein the first module public key is received before the second module public key, and wherein the second module public key is authenticated using the first module public key.

**8**. The method of claim **1**, further comprising:

receiving the module instruction from an application server;

decrypting the module instruction using a first server public key, wherein the first server public key is processed using an RSA algorithm; and,

ciphering the server encrypted data using a second server public key, wherein the second server public key is processed using an ECC algorithm.

**9**. The method of claim **1**, wherein the module identity in the first message comprises a first string, and wherein the module identity in the second message comprises a second string, and wherein the first string and the second string are associated with a serial number of a module.

**10**. The method of claim **1**, wherein ciphering the server encrypted data using a module public key from the series further comprises ciphering the server encrypted data with a symmetric key, wherein the symmetric key is shared by using at least one of (i) the module public key from the series and a key derivation function, (ii) an asymmetric ciphering algorithm and the module public key from the series to encrypt the symmetric key, wherein the set of servers sends the symmetric key, and (iii) a digital signature verified with the module public key from the series, wherein the set of servers receives the symmetric key and the digital signature.

**11**. A method for supporting machine-to-machine communications, the method performed by a set of servers, the method comprising:

using a first server private key to establish a secure connection with at least one application server;

receiving a message, wherein the message includes a module identity and a module digital signature, wherein the module digital signature is verified using a first module public key;

sending a response, wherein the response includes a server digital signature processed using a second server private key;

sending a set of cryptographic parameters for deriving a second module public key;

receiving the second module public key, wherein the second module public key is verified using the first module public key, wherein the second module public key is used to decrypt a module encrypted data, and wherein the module encrypted data includes a sensor data; and,

sending the sensor data and the module identity to the application server.

**12**. The method of claim **11**, wherein the module identity in the first message comprises a first string, and wherein the module identity sent with the sensor data comprises a second string, and wherein the first string and the second string are associated with a serial number of a module.

**13**. The method of claim **11**, wherein the set of servers includes at least a first server and a second server, wherein (i) the first server receives the message and sends the response, and (ii) the second server receives the module encrypted data, wherein the set of servers includes a shared module database, wherein the shared module database records the module identity, the first module public key, and the second module public key.

**14**. The method of claim **11**, wherein the set of cryptographic parameters includes at least one of (i) an elliptic curve cryptography (ECC) named curve, (ii) a value for an ECC defining equation, (iii) a modulus for an RSA algorithm, (iv) a time-to-live value, (v) a value for a random number seed, (vi) a value for a module public key identity, (vii) a value for a module random seed file.

**15**. The method of claim **11**, wherein a module for the module identity sends the message and the second module public key, and wherein the module derives the second module public key using at least a set of cryptographic algorithms and the set of cryptographic parameters.

**16**. The method of claim **15**, wherein the module comprises at least one of a mobile phone and a payment terminal, wherein one of the mobile phone and the payment terminal includes a universal integrated circuit card (UICC), wherein the UICC records a module private key associated with the first module public key, and wherein the set of servers records the first module public key and the module identity.

**17**. The method of claim **11**, wherein the first and second server private keys are different, wherein the first server private key is processed using an RSA algorithm and the second server private key is processed using an ECC algorithm.

**18**. The method of claim **11**, further comprising:

receiving a module instruction and the module identity; and,

waiting until after receiving a third message before sending the module instruction within a server encrypted data, wherein the server encrypted data is ciphered using the second module public key.

**19**. The method of claim **11**, wherein decrypting the module encrypted data using the second module public key further comprises decrypting the module encrypted data with a symmetric key, and at least one of (i) sending the symmetric key using an asymmetric ciphering algorithm and the second

module public key, (ii) deriving the symmetric key using a key derivation function and the second module public key, and (iii) receiving (a) the symmetric key using an asymmetric ciphering algorithm and (b) a digital signature, wherein the digital signature is verified using the second module public key.

**20**. The method of claim **11**, wherein verifying the second module public key using the first module public key comprises at least one of (i) receiving the second module public key with a digital signature, where the digital signature is verified with the first module public key, and (ii) receiving an encrypted data with the second module public key, wherein the encrypted data is decrypted using the first module public key.

**21**. A system for supporting machine-to-machine communications, the system comprising:

A module controller for monitoring a first Internet protocol address and port (IP:port) number, for receiving a first message comprising a module public key, a module identity, a set of cryptographic parameters, and a first source IP:port number, for sending a server digital signature, for receiving a second message comprising the module identity and a second source IP:port number, and for sending a response to the second message, wherein the response comprises (i) a module instruction and (ii) the second source IP:port number as a destination IP:port number;

An application interface for monitoring a second IP:port number, for receiving a module instruction and the module identity, wherein the module instruction with the module identity is received after the first message and before the second message;

A shared module database for recording a shared secret key, the module identity, the module public key, and the set of cryptographic parameters; and,

A set of cryptographic algorithms for using the shared secret key to verify the received module public key is associated with the recorded module identity, and for processing the server digital signature using a server private key.

**22**. The system of claim **21**, further comprising:

a set of servers, wherein the set of servers includes the module controller and the application interface, wherein the application interface uses a first IP address to receive the application message and the module controller uses a second IP address to receive the second message; and,

the cryptographic algorithms for decrypting at least one of the first module public key and the set of cryptographic parameters.

**23**. The system of claim **21**, wherein verifying the module public key using the shared secret key comprises using the shared secret key and at least one of (i) a message digest algorithm, (ii) a module digital signature, wherein the module digital signature was processed using the shared secret key, and (iii) a symmetric ciphering algorithm, wherein the symmetric ciphering algorithm uses the shared secret key.

**24**. The system of claim **21**, further comprising the set of cryptographic algorithms for using a first server public key with the application interface, and a second public key with the module controller, wherein the first server public key is associated with an RSA algorithm, and wherein the second public key is associated with an elliptic curve cryptography (ECC) algorithm.

**25**. The system of claim **21**, wherein the set of cryptographic parameters includes at least one of (i) an elliptic curve cryptography (ECC) named curve, (ii) a value for an ECC defining equation, (iii) a modulus for an RSA algorithm, (iv) a time-to-live value, and (v) a value for a module public key identity.

**26**. The system of claim **21**, wherein the second message includes a server instruction, wherein the server instruction is received in a module encrypted data, wherein the module encrypted data includes a security token, and at least one of a sensor measurement, a registration message, and an acknowledgement, and wherein the module encrypted data is processed using the set of cryptographic parameters.

**27**. The system of claim **21**, further comprising the module controller for receiving a series of different module public keys with the module identity after receiving the first message, wherein each of the different module public keys in the series is verified using a prior module public key in the series.

* * * * *