

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

Appendix 838-02

The following list illustrates prior art references for invalidity:

- S. Bugiel, et al., “TruWalletM: Secure Web Authentication on Mobile Platforms,” Second International Conference on Trusted Systems (INTRUST 2010), pp. 219-36 (Dec. 13-15, 2010) (“*Bugiel*”), which was published and has been publicly available since December 13-15, 2010, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. § 102(a) and possibly under pre-AIA § 102(b).
- U.S. Patent Application Publication No. 2011/0191599 A1 to Chou et al (“*Chou*”), which was filed on February 26, 2010, and published August 4, 2011, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a) and (e).
- U.S. Patent Application Publication No. 2006/0107032 A1 to Paaske et al (“*Paaske*”), which was filed on October 13, 2005, and published May 18, 2006, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a), (b) and (e).
- U.S. Patent Application Publication No. 2010/0296651 to Tkacik (“*Tkacik*”), which was filed on May 21, 2009, and published November 25, 2010, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a), (b) and (e).
- U.S. Patent No. 6,996,547 to Tugenberg et al. (“*Tugenberg*”), which was filed on September 27, 2000, and issued on February 7, 2006, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a), (b) and (e).
- N. Aaraj, et al, “Analysis and Design of a Hardware/Software Trusted Platform Module for Embedded Systems,” ACM Transactions on Embedded Computing Systems, vol. 8, no. 1, art. 8, pp. 1-31 (December 2008) (“*Aaraj*”), which was published and has been publicly available since at least January 4, 2009, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a) and (b).
- U.S. Patent Application Publication No. 2011/0314271 A1 to Boccon-Gibod et al. (“*Gibod*”), which was filed on Jun 17, 2011, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. § 102(e).
- U.S. Patent Application Publication No. 2011/0258426 to Mujtaba et al (“*Mujtaba*”), which was filed on November 11, 2010, and published on October 20, 2011, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a) and (e).
- U.S. Patent No. 7,773,754 to Buer et al (“*Buer754*”), which was filed on July 8, 2002, and issued on August 10, 2010, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a), (b) and (e).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

- T. Dierks & E. Rescorla, “The TLS Protocol Version 1.2,” Internet Engineering Task Force (IETF), Network Working Group Request for Comments No. 5246 (August 2008) (“*TLS-1.2*” also known as RFC 5246), which was published and has been publicly available since at least August 2008, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a) and (b).
- T. Dierks & E. Rescorla, “The TLS Protocol Version 1.1,” Internet Engineering Task Force (IETF), Network Working Group Request for Comments No. 4346 (April 2006) (“*TLS-1.1*” also known as RFC 4346), which was published and has been publicly available since at least April 2006, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a) and (b).
- T. Dierks & C. Allen, “The TLS Protocol Version 1.0,” Internet Engineering Task Force (IETF), Network Working Group Request for Comments No. 2246 (January 1999) (“*TLS-1.0*” also known as RFC 2246), which was published and has been publicly available since at least January 1999, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a) and (b).
- A. Freier et al., “The SSL Protocol Version 3.0,” Internet Engineering Task Force (IETF), Network Working Group Request for Comments No. 6101 (August 2011) (“*TLS-3.0*” also known as RFC 6101), which was originally published by Netscape Communications Corporation November 18, 1996, and has been publicly available since then,¹ qualifies as prior art under at least pre-AIA 35 U.S.C. §§ 102(a) and (b).
- C. Kaufman et al., “Internet Key Exchange Protocol Version 2 (IKEv2),” Internet Engineering Task Force (IETF), Network Working Group Request for Comments No. 5996 (September 2010) (“*RFC-5996*”), which was published and has been publicly available since at least September 2010, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a) and (b).

¹ The Internet Engineering Task Force (IETF) also published a historical record of the SSL 3.0 protocol in August 2011. *See* A. Freier, et al., “The Secure Socket Layer (SSL) Protocol Version 3.0,” IETF Request for Comments 6101 (August 2011). This document states in its Foreword that “Although the SSL 3.0 protocol is a widely implemented protocol, a pioneer in secure communications protocols, and the basis for Transport Layer Security (TLS), it was never formally published by the IETF, except in several expired Internet-Drafts. This allowed no easy referencing to the protocol. We believe a stable reference to the original document should exist and for that reason, this document describes what is known as the last published version of the SSL 3.0 protocol, that is, the November 18, 1996, version of the protocol.”

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

- C. Kaufman, “Internet Key Exchange (IKEv2) Protocol,” Internet Engineering Task Force (IETF), Network Working Group Request for Comments No. 4306 (December 2005) (“*RFC-4306*”), which was published and has been publicly available since at least December 2005, qualifies as prior art to the '838 Patent under at least pre-AIA 35 U.S.C. §§ 102(a) and (b).

As illustrated below, to the extent that *Bugiel* is found not to anticipate, expressly or inherently, one or more of the asserted claims of the '838 Patent, each of claims 1-24 are invalid as obvious in view of *Bugiel* in combination with other prior art references, including, but not limited to, the prior art discussed below, the prior art identified in Defendant's Invalidity Contentions, and the prior art described in additional claim charts attached as Appendices 838-01, 838-03 through 838-06. Exemplary invalidity grounds based on the *Bugiel* reference include the following, although these examples are by no means limiting:

- Obviousness based on *Bugiel* (claims 1-24);
- Obviousness based on *Bugiel* in combination with any of the SSL/TLS protocols (*SSL-3.0*, *TLS-1.0*, *TLS-1.1*, or *TLS-1.2*) (claims 1-24);
- Obviousness based on *Bugiel* in combination with any of the IPsec IKE protocols (*RFC-5996* or *RFC-4306*) (claims 1-2, 5-10, 13-15, 17-18, 21-24);
- Obviousness based on *Bugiel* in combination with *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754* (claims 1-24)
- Obviousness based on *Bugiel* in combination with *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*; and any of the SSL/TLS protocols (*SSL-3.0*, *TLS-1.0*, *TLS-1.1*, or *TLS-1.2*) (claims 1-24); and
- Obviousness based on *Bugiel* in combination with *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*; and any of the IPsec IKE protocols (*RFC-5996* or *RFC-4306*) (claims 1-2, 5-10, 13-15, 17-18, 21-24).

Further, Defendant expressly incorporates by reference the grounds and arguments set forth in *MediaTek Inc. v. Daedalus Prime LLC*, No. IPR2025-00100, Paper 1 (P.T.A.B. November 15, 2024), and Exhibits 1001-1025 to the same, including all relevant motivation to combine rationales.

Further, Defendant expressly incorporates by reference the grounds and arguments set forth in *Samsung Electronics Inc. et al. v. Daedalus Prime LLC*, No. IPR2023-00727, Paper 3 (P.T.A.B. March 20, 2023), and Exhibits 1001-1011 to the same, including all relevant motivation to combine rationales.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The citations included in the table below are exemplary only and are not intended to be limiting or exhaustive. Defendant reserves the right to cite additional disclosures and evidence in support of their positions as this case develops further. Additionally, citation to a particular figure in a reference should be understood to encompass the caption and description of the figure and any text relating to or discussing the figure. Similarly, citation to any text referring to a figure should be understood to include the figure as well. Additionally, Defendant may be basing the arguments below on one or more claim construction positions that Daedalus has not expressly disclosed but seems to apply in its Patent Local Rule 3-1 disclosures or Defendant's present understanding of Daedalus's view and application of claim scope. By applying the prior art in this way, Defendant does not concede its agreement with Daedalus's positions, nor do these Invalidity Contentions constitute any admission regarding claim scope. Nor is Defendant asserting any claim construction positions through these charts. Moreover, claim preambles generally are not limiting, and nothing in these contentions should be deemed an admission that the preamble in the asserted claims is limiting. As this case further develops, Defendant may amend or supplement these exemplary disclosures and any related contentions as appropriate, and in a manner consistent with all applicable rules governing this case. In particular, Defendant reserves the right to rely on (1) uncited portions of the identified prior art; (2) other prior art not identified herein; (3) references that show the state of the art (irrespective of whether such references themselves qualify as prior art to the asserted patents); (4) factual testimony from the inventors or authors of the prior art references; and/or (5) expert testimony, to provide context to or aid in understanding the prior art and the state of the art at the time of the alleged inventions. Defendant also reserves the right to amend or supplement these exemplary disclosures based on changes in Daedalus's positions as the case progresses, including changes to Daedalus's infringement theories, claim interpretations, or apparent claim constructions.

Descriptions of Exemplary Obviousness Combinations Based on *Bugiel*

Bugiel and each of the additional references cited below are directed to device architectures with similar structures and functions. They are analogous art because they come from the same field of endeavor as the '838 patent and are reasonably pertinent to the problems the '838 patent addresses. Because these references are all directed to the same types of devices and have similar structures and functions, a person of ordinary skill in the art ("POSITA") would have appreciated that *Bugiel* and the additional references cited below provide disclosures that relate to and complement one another. Moreover, a POSITA would have found it obvious to combine their disclosures in the ways discussed below.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

Exemplary Reasons for Implementing *Bugiel* in a System-on-a-Chip (SoC) Apparatus as Claimed

Bugiel expressly teaches using secure hardware for mobile devices. *See generally Bugiel* at 219-230, 232-236. More specifically, *Bugiel* teaches to implement its design using any “secure hardware for mobile devices” that provides “a Trusted Execution Environment (TrEE) to protect the execution of critical code from tampering.” *Id.* at 220; *see also id.* at 234 (“Our solution does not rely on a secure operating system, but exploits hardware security features, in particular Trusted Execution Environment (TrEE) available on many modern mobile phones.”).

A POSITA also would have understood the IKE or SSL/TLS processes *Bugiel* and each of the IKE (*RFC-5996* and *RFC-4306*) or SSL/TLS protocols (*SSL-3.0*, *TLS-1.0*, *TLS-1.1* and *TLS-1.2*) describe hardware components, including one or more of the following, e.g., Mobile device with baseband and transceivers for communication with a remote server. *See Bugiel* at 221 (“We consider application scenarios where a user deploys his mobile device to access services provided by remote web servers (over the Internet).”); secure storage and isolated secure execution environment. *See Bugiel* at 222 (“Assumptions. We rely on availability of a secure hardware which provides a Trusted Execution Environment (TrEE) with following features: (i) isolated secure code execution, (ii) secure storage, (iii) integrity protection of secure execution environment”), 232 (“the TrEE is isolated from the rest of the system in terms of processing and memory”); performing cryptographic functions and key management functions, such as generating random nonces, implementing Diffie-Hellman, RSA, hashing, signing, or encryption/decryption. *See TLS-1.2* at 4, 34, 46, 52, 62, 65; *see also TLS-1.1* at 4, 11, 31-32, 50-53; *TLS-1.0* at 3, 29-30, 45-56, 48; *SSL-3.0* at 18, 30-33; encrypting session key with security key specific to the device. *See Bugiel* at 225, 231.

Each of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754* describes relevant secure hardware for mobile devices as described in *Bugiel*.

For example, *Chou*’s HSM (Hardware Security Module) 20 establishes a secure area within an SoC 91 and provides a secure zone for a mobile device—the type of hardware *Bugiel* teaches to use. *See Chou* ¶ [0018] (“[A] HSM 20 to set a secure domain, zone or boundary to protect a secure key. The term ‘secure’ (such as secure domain, zone or boundary) is used herein to designate a zone controlled by HSM 20. Thus, components outside of the zone established by HSM 20 reside in a non-secure zone relative to HSM 20.”), ¶ [0027] (“HSM 20 ensures that any secure key and/or secure context data that is unwrapped remains within HSM 20 and, further, any unauthorized access or copying from outside of HSM 20 is prevented.”); ¶ [0036] (“That is, all accesses in and out of HSM 20 are routed via IPC/Msg RAM module 61 when in the secure mode of operation, in order to control accesses to HSM 20 by non-secure components that reside outside of HSM 20.”), ¶ [0068] (“[D]evice 90 is implemented in a mobile phone 102, such as a cell phone.”), Figs. 1, 3. A POSITA thus would have found it obvious to implement *Bugiel*’s system using hardware that includes *Chou*’s HSM 20.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

Further, a POSITA would have understood that both *Chou* and *Bugiel* relate to secure communication using cryptographic keys for encryption and decryption. For example, *Bugiel* states, “For secure communication between the device and the remote web server we utilize SSL/TLS protocol,” and also explicitly references *TLS-1.2* regarding how to implement the SSL/TLS handshake. *See Bugiel* at 222; *see also id.* at 225 (“All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16].”²). *Chou* also describes secure communications in which “secure data transfer is utilized to perform secure financial transactions over a network, such as the Internet. In a typical secure transaction, data content that contains financial information, such as a credit/debit card information, are encrypted and transmitted to a designated recipient through an unsecure network. The recipient of the secure data utilizes a secure key to decrypt the data to retrieve the financial information.” *Chou* ¶ [0006]. *Chou* also explains the application of its hardware “to a mobile phone,” *id.* ¶ [0017], and further describes a “technique for providing secure key operation, in which a secure zone boundary is defined by components that make up a hardware security module (HSM),” *id.* ¶ [0069]; *see also Chou* at ¶ [0009] (“secure key management scheme”), ¶¶ [0017]-[0018], ¶ [0039], ¶¶ [0024]-[0025], ¶¶ [0040]-[0041], ¶¶ [0043]-[0054].

A POSITA also would have understood *Bugiel*, as well as each of the IKE (*RFC-5996* and *RFC-4306*) or SSL/TLS protocols (*SSL-3.0*, *TLS-1.0*, *TLS-1.1* and *TLS-1.2*) describe one or more of the following hardware components, e.g., mobile device with baseband and transceivers for communication with a remote server. *See Bugiel* at 221 (“We consider application scenarios where a user deploys his mobile device to access services provided by remote web servers (over the Internet).”); secure storage and isolated secure execution environment. *See Bugiel* at 222 (“Assumptions. We rely on availability of a secure hardware which provides a Trusted Execution Environment (TrEE) with following features: (i) isolated secure code execution, (ii) secure storage, (iii) integrity protection of secure execution environment”), 232 (“the TrEE is isolated from the rest of the system in terms of processing and memory”); performing cryptographic functions or key management functions, such as generating random nonces, implementing Diffie-Hellman, RSA, hashing, signing, or encryption/decryption. *See TLS-1.2* at 4, 34, 46, 52, 62, 65; *see also TLS-1.1* at 4, 11, 31-32, 50-53; *TLS-1.0* at 3, 29-30, 45-56, 48; *SSL-3.0* at 18, 30-33; encrypting session key with security key specific to the device. *See Bugiel* at 225, 231.

Chou explains that its SoC apparatus includes components that are capable of performing relevant cryptographic functions, such as SoC 91, would be implemented in a wireless device 90 used to transmit and receive wireless communication, where the “wireless device 90 includes a baseband processor 93 and radio 94” (which includes transceivers). *Chou* ¶ [0066], Fig. 5. In addition, *Chou* explains that HSM 20 of SoC 91 sets a secure zone that separates HSM and SPU-M processing from CPU 11/32. *See Chou* ¶ [0018] (“HSM 20 to set a secure domain, zone or boundary to protect a secure key. The term ‘secure’ (such as secure domain, zone or boundary) is used herein to designate a zone controlled by HSM 20.”), Figs. 1, 3. *Chou* explains that HSM 20 and SPU-M includes secure memories, such as secure key module 21, secure memory 22, OTP 54, key cache 65, HSM ROM 52, HSM RAM 53, etc., that are not accessible by

² Reference [16] of *Bugiel* is TLS version 1.2, i.e., *TLS-1.2*. *See Bugiel* at 236 (“16. Network Working Group: The transport layer security (TLS) protocol. version 1.2. Standards track (2008) <http://tools.ietf.org/html/rfc5246>.”).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

components outside the HSM and SPU-M, thus ensures the contents (such as secrets/keys, random nonces, etc.) stored in the secure memory is protected. *See Chou* ¶ [0063] (“[A]ll accesses and transfers to and from HSM 20 to/from components outside of HSM 20 are routed through IPC/Msg RAM module 61. In this way, access to retrieve and/or copy unwrapped secure keys in HSM 20 are prevented and secure keys never leave HSM 20 in an unwrapped state.”); *see also* ¶ [0018], ¶¶ [0020]-[0025], ¶ [0027], ¶¶ [0030]-[0031], ¶¶ [0036]-[0037], ¶¶ [0040]-[0041], ¶ [0055], ¶ [0069], Figs. 1, 3. *Chou* discloses a RNG (random number generator) module 58 within HSM 20 to generate random values “as a seed for a cryptographic algorithm.” *Chou* ¶ [0042]. *Chou* explains that HSM 20 includes a PKA (Public Key Accelerator) module 56, which is an accelerator operable for the cryptographic key exchange algorithms in IPsec IKE or SSL/TLS protocols, such as Diffie-Hellman public value and shared key computation and RSA key generation. *See Chou* ¶¶ [0043]-[0053]. *Chou*’s PKA module is also operable for signing and hashing algorithms like “1024-bit DSA signature with SHA-1 hash or 2048-bit DSA signature with SHA-256 hash.” *See Chou* ¶ [0045]. *Chou* explains that SoC 91 includes a Key Algorithm module/SPU-M (Secure Processing Unit for Mobile processors) that operates as a hardware accelerator for symmetric key algorithms, “such as AES, DES, RC4, SHA1-HMAC, SHA2-HMAC, MD5-HMAC.” etc., for encrypting/decrypting messages using key from key cache 65. *Chou* ¶¶ [0054]-[0055]. *Chou* explains that HSM 20 includes a KEK (key-encryption key) unit 55, which wraps and unwraps keys using a root/device unique key programmed in OTP during manufacturing. *See Chou* ¶ [0023], ¶ [0040], ¶ [0064]. Therefore, *Chou* discloses a device unique key/root key that is never available outside of secure domain 80 and can be used to encrypt security keys such as a session key or pre-master key, etc. *See also Chou* ¶ [0003], ¶ [0007], ¶ [0012], ¶ [0018], ¶ [0021], ¶¶ [0023]-[0027], ¶¶ [0034]-[0035], ¶¶ [0037]-[0041], ¶ [0054], ¶ [0064], ¶¶ [0069]-[0071], claims 1-3, 6-7, 12, 14, 17-18, Abstract, Figs. 1, 3.

Because *Chou* teaches an apparatus with hardware components that are relevant to the functions described by *Bugiel* and/or IPsec IKE or SSL/TLS protocols, a POSITA would have found it obvious to implement the techniques described in *Bugiel*, using *Chou*’s SoC apparatus. *See Chou* ¶ [0017] (“The embodiments of the present invention may be practiced in a variety of settings that utilize a secure key to authenticate and unwrap and/or decrypt data, context data and/or a secure key. The described embodiments below pertain to a particular hardware security module (HSM), but other embodiments may have other name designations. Furthermore, the application of the described embodiment pertains to a mobile phone, but the invention need not be limited to mobile communication or other wireless applications. The invention may be utilized in wired settings, such as wired networks, or other environments having physical conductive connections. The invention is applicable in a setting where secure keys are employed and accesses to the key by unauthorized circuits, components, devices, software, firmware, etc. are to be controlled.”),

The combinations based on *Chou* alleged here incorporate *Chou*’s hardware to implement *Bugiel*’s methods in TrEE, which includes implementing known protocols e.g., the IPsec IKE or the SSL/TLS Handshake Protocol, etc., to establish a secure communication session. For example, the alleged combinations use *Chou*’s HSM 20 and SPU-M 39 to perform the cryptographic key exchanges, generate and use keys based on the IKE or SSL handshake protocol, encrypt keys using *Chou*’s security key, and store relevant secure parameters (such as encrypted secrets/keys, random nonces, DH values, etc.) in its secure memory for protection. *See Chou* ¶¶ [0042]-

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

[0055], ¶ [0018], ¶ [0066] (describing the capabilities of its hardware components as discussed above), ¶¶ [0069]-[0071]; *Bugiel* at 221-222, 225-227, 231 (describing various functions performed in its design).

In addition, *Tkacik* also relates to the same technical area of performing secure communication protocols and key management as *Bugiel*. See *Tkacik* ¶¶ [0006]-[0009] (“managing secret keys”), ¶¶ [0001]-[0009], ¶ [0019], ¶ [0021], ¶¶ [0029]-[0031], ¶¶ [0037]-[0039], ¶¶ [0049]-[0053], ¶¶ [0055]-[0056], ¶¶ [0064]-[0089]. Relevant to *Bugiel*’s secure execution environment on SoC, Figure 1 of *Tkacik* illustrates a secure execution environment such as the secure processing section 30, which is separate from programmable processor 26 of the SoC secure processing system 12; see also *Tkacik Abstract*, ¶¶ [0001]-[0002], ¶¶ [0005]-[0008], ¶¶ [0012]-[0013], ¶¶ [0015]-[0018], ¶ [0020], ¶¶ [0022]-[0026], ¶¶ [0029]-[0034], ¶ [0037], ¶¶ [0039]-[0040], ¶¶ [0043]-[0044], ¶¶ [0046]-[0049], ¶ [0058], ¶ [0064], ¶¶ [0066]-[0067], ¶¶ [0073]-[0074], ¶ [0079], ¶ [0087], ¶ [0089], Figure 2. In addition, *Tkacik* describes its hardware is capable of performing key generation using any of a number of protocols known to those skilled in the art, including IKE, DH, RSA key gen., SSL, random key, and the like.” *Tkacik* at ¶¶ [0079]-[0080]. *Tkacik* also describes its hardware is capable of performing the various wireless and cryptographic functions (generating random nonce, signing, hashing, encryption/description, RSA, Diffie-Hellman, IKE, SSL, etc.) relevant to *Bugiel*, *IPsec IKE* or *SSL/TLS* protocols. See *Tkacik Abstract*, ¶¶ [0002]-[0009], ¶¶ [0019]-[0021], ¶ [0023], ¶ [0026], ¶¶ [0028]-[0029], ¶ [0031], ¶¶ [0037]-[0038], ¶ [0045], ¶¶ [0047]-[0056], ¶¶ [0059]-[0062], ¶¶ [0064]-[0089], Abstract, Figs. 1-3, 5-8.

In addition, relevant to *Bugiel*’s secure memory, *Tkacik* describes its secure processing section 30 includes secure memory that is only accessible by the secure processing section. More specifically, Figure 2 of *Tkacik* illustrates the following secure memories are located within the secure processing section 30 and only accessible by the secure processing section: e.g., zeroizable read-write memory 52, key register 54, long-lived non-volatile key register 60, active key register 72, control register 74, data register 76, etc. See *id.* ¶¶ [0036]-[0038], ¶¶ [0044]-[0046], ¶¶ [0054]-[0055], Fig. 2. *Tkacik* also explains that key register 54, active key register 72, control register 74, and data register 76 can be included as part of read-write memory 52. See *id.* ¶ [0038], ¶ [0054]. These memories all reside within the secure processing section 30. Therefore, these memories are the claimed “secure memory accessible only by” the secure processing section 30 (the claimed “security engine”); see also *id.* ¶¶ [0005]-[0008], ¶¶ [0036]-[0039], ¶¶ [0041]-[0042], ¶¶ [0044]-[0049], ¶¶ [0052]-[0055], ¶ [0059], ¶¶ [0067]-[0068], ¶¶ [0070]-[0071], ¶¶ [0074]-[0075], ¶ [0077], ¶¶ [0080]-[0084], ¶ [0087], ¶ [0089], claims 1-2, 8-9, 10-11, 13-14, 16, 18, Figure 2. *Tkacik* further explains contents in these memories “are protected against the execution of malicious software within host device 10.” *Tkacik* ¶ [0037], ¶ [0029]. In addition, the security processing section is design to detect tamper attempt for keeping the operations and data within the secure processing section safe. See *Tkacik* ¶¶ [0039]-[0043], ¶ [0045], ¶ [0061], ¶ [0069], ¶ [0072]. This is consistent with *Bugiel*’s assumption of secure storage for keeping the sensitive data secure. *Bugiel* at 222-225, 228, 231-234. Therefore, a POSITA would have found it obvious to implement *Bugiel*’s secure memory as *Tkacik*’s secure memory.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

Further, relevant to *Bugiel*'s security key, *Tkacik* discloses a long-lived key register 60 (part of the claimed "secure memory") includes a plaintext master key 160 (the claimed "security key") that was encoded in the long-lived key register 60 during a manufacturing process of the SoC secure processing system 12. *See Tkacik* ¶ [0046] ("long-lived key register 60 is desirably programmed with plaintext master key 160 during the manufacturing process of secure processing system 12 and cannot thereafter be altered."); *see also* ¶¶ [0002]-[0004], ¶¶ [0006]-[0009], ¶ [0019], ¶ [0029], ¶ [0039], ¶¶ [0044]-[0045], ¶ [0048], ¶ [0057], ¶¶ [0069]-[0070], ¶ [0074], ¶¶ [0083]-[0085], ¶¶ [0088]-[0089], Figure 2, Abstract. *Tkacik* also describes the secure processing section uses this master key to encrypt other keys generated by the secure processing section. For example, *Tkacik* explains the plain master key 160 can "directly serve[] as the BKEK" if desired. *Id.* ¶ [0083]. Here, BKEK means "blob key encryption key." *Id.* ¶ [0069]. The BKEK is a cryptographic key to be used in forming a blob, "a self-contained object which includes a cryptographic key in an encrypted form and a message authentication code (MAC)." *Id.* ¶ [0070]. *Tkacik* further teaches generating a plaintext active key 172 using IKE or SSL. *Tkacik* ¶ [0080]. Referring to process 140 in *Tkacik*'s Figure 8, *Tkacik* describes moving the generated plaintext active key 172 "to data register 76 and encrypt[ing] this key 172 using . . . the BKEK" (the plaintext master key 160, i.e., the claimed "security key"). *Tkacik* ¶ [0084], Fig. 8. According to *Tkacik*, the secure process section 30 performs this encryption. *See Tkacik* ¶ [0079] ("Process 140 may be performed by controller 46 in response to the execution of software programming instructions using components of secure processing section 30, and possibly other components of host device 10."). *Tkacik*'s description of using the master key to encrypt other keys generated based on IKE, SSL or other similar protocol is consistent with *Bugiel*'s description of using a key to encrypt the session key and pre-master key. *See Bugiel* 225-227, 231; *see also Tkacik* ¶¶ [0002]-[0004], ¶¶ [0006]-[0009], ¶ [0019], ¶ [0029], ¶ [0039], ¶¶ [0044]-[0046], ¶¶ [0049]-[0054], ¶ [0057], ¶¶ [0069]-[0070], ¶ [0074], ¶¶ [0083]-[0085], ¶¶ [0088]-[0089], Figure 2, Abstract.

Therefore, a POSITA would have found it obvious to implement *Bugiel*'s method in *Tkacik*'s SoC apparatus. The combinations based on *Tkacik* alleged here incorporate *Tkacik*'s SoC hardware to implement *Bugiel*'s methods in secure execution environment, which includes implementing known protocols, e.g., the IKE or the SSL/TLS Handshake Protocol, etc. to establish a secure communication session. For example, the alleged combinations use *Tkacik*'s secure processing section to perform the cryptographic key exchanges, generate and use keys based on the IKE or SSL handshake protocol, encrypt keys using *Tkacik*'s security key, and store relevant secure parameters (such as encrypted secrets/keys, random nonces, DH values, etc.) in its secure memory for protection. *See Bugiel* at 221-222, 225-227, 231 (describing various functions performed in its *Bugiel*'s design).

In addition, *Paaske* also relates to the same technical area of performing secure communication and key management in a mobile device as *Bugiel*. *See Paaske* ¶ [0003], ¶ [0032] ("enable the computing system to communicate with external systems via, for example, a wired or wireless network or a removable media via an external connection 212."). Relevant to *Bugiel*'s secure execution environment on SoC, Figure 2 of *Paaske* illustrates integrated circuit 204 (a "system-on-a-chip") comprising a TPM 202 and cryptographic processor(s) 220 that are separate from other processing core 222 on the SoC. *See Paaske* ¶ [0032] ("[T]he TPM 202 is implemented in an integrated circuit 204."), ¶ [0034] ("In some embodiments the integrated circuit 204 may include other components to, for example, reduce the number of integrated circuits on the motherboard. For example, one or more cryptographic processor(s) 220 that may be used by the

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

TPM 202 may be incorporated into the integrated circuit. In addition, the integrated circuit 204 may include one or more other processing component(s) 222 that may operate independently of the TPM 202.”); *see also Paaske* at Abstract, Figs. 2-3. *Paaske* also teaches that cryptographic processors 220 may be included within the TPM. *See id.* ¶ [0031] (“[T]he TPM may include one or more cryptographic processors.”). For example, *Paaske*’s Figure 3 shows cryptographic processors 330 and 334 as being included within TPM 302. *See also id.* ¶ [0054] (“[T]he instruction cache controller 326 may include one or more cryptographic processor(s) 334 for performing the decryption and authentication operations.”), ¶ [0055] (“The cryptographic processor(s) 330 may be used to provide cryptographic processing for the TPM 302. For example, the cryptographic processor(s) 330 may include a public key core 350 . . . an authentication core 352 . . . [and] a symmetric cryptographic core 354 . . .”). For purposes of this chart, the TPM may include such cryptographic processors and that the TPM itself (e.g., TPM 302) may constitute the claimed “security engine.”; *see also Paaske* ¶ [0007], ¶ [0050], ¶ [0111], ¶ [0227]. In addition, *Paaske* also describes its security engine may perform cryptographic functions relevant in *Bugiel* or IKE or SSL/TLS protocols. *See Paaske* at ¶ [0005] (encryption/decryption), ¶¶ [0028]-[0029] (encryption/decryption, authentication), ¶ [0031] (encryption, decryption, authentication, verification, signing and key management), ¶¶ [0042]-[0044] (encryption, authentication), ¶¶ [0055]-[0056] (DSA, RSA, Diffie-Hellman, hashing, symmetric encryption), ¶ [0101] (encryption, signing), ¶ [0106] (RSA), ¶ [0114] (encryption and authentication), ¶ [0154] (random number generator), ¶ [0169] (RSA, digital signing), ¶ [0177] (RSA, digital signing), ¶ [0191] (random number generator), ¶ [0194] (RSA), ¶ [0175] (hash), ¶ [0105] (hash), ¶ [0131] (hash), ¶ [0156] (hash), ¶¶ [0205]-[0206] (RSA), ¶¶ [0208] –[0210] (RSA); ¶¶ [0217]-[0227], Figs. 2 and 3.

In addition, relevant to *Bugiel*’s secure memory, *Paaske*’s Figure 3 indicates that all of these memories reside within the TPM (302) and the cryptographic processors and thus can be accessible only by the TPM and cryptographic processors. Collectively, these memories are the claimed “secure memory” accessible only by the TPM and the cryptographic processors. *Paaske* also describes the TPM provides “secure protection” of the secure execution environment. *See Paaske* ¶ [0007]-[0008], ¶ [0056]. This is consistent with *Bugiel*’s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. *Bugiel* at 222-225, 228, 231-234.

In addition, relevant to *Bugiel*’s security key, *Paaske* teaches a security key by disclosing that “when the TPM 302 is manufactured security routines may be used to generate secret keys that are unique to each TPM. These keys may then be stored in the OTP memory 332. Moreover, provisions may be made to ensure that these secret keys are not allowed to be read out of the TPM (or the integrated circuit within which the TPM resides) in the clear (e.g., in unencrypted form).” *Paaske* ¶ [0056]; *see also id.* ¶ [0101] (“[A]t some point during the manufacturing process . . . an encryption key . . . may be generated for the code update process.”), ¶ [0108] (“[A]t some point during the manufacturing process (e.g., during a test or boot procedure) the TPM may generate . . . internal keys that may be unique to that specific TPM chip.”). Therefore, *Paaske* discloses the claimed “security key” by describing “secret keys” that are generated when the TPM is manufactured and stored in the OTP 332 (part of the claimed “secure memory”). *See Paaske* at ¶¶ [0043]-[0044], ¶¶ [0110]-[0111], ¶¶ [0059]-[0060], ¶¶ [0217]-[0227], Fig. 3.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

Paaske explains that its secret keys are used for encrypting other keys. See *Paaske* ¶ [0056] (“Through the use of these keys and/or keys derived from these keys, the TPM may securely store information outside of the TPM (or the integrated circuit within which the TPM resides). For example, the cryptographic processor(s) 330 may use these keys to encrypt and/or authenticate any data that is stored in an external memory such as the flash memory 306.”), ¶ [0110] (“[T]he keys stored in the OTP memory may be used to encrypt other keys so that these other keys may be securely stored in external memory.”). *Paaske* also explains its secret keys are protected in the OTP of the TPM. Therefore, *Paaske* teaches a security key (e.g., a unique internal security key) that was encoded in the secure memory (e.g., in OTP 332) during a manufacturing process of the system-on-a-chip (e.g., *Paaske*’s integrated circuit). *Paaske*’s description of using key that is not allowed to be read out of the TPM to encrypt other keys is consistent with *Bugiel*’s description of using a key that is never available outside TrEE to encrypt the session key and pre-master key. See *Bugiel* 225-227, 231.

Therefore, a POSITA would have found it obvious to implement *Bugiel*’s method in *Paaske*’s SoC apparatus. The combinations based on *Paaske* alleged here incorporates *Paaske*’s SoC hardware to implement *Bugiel*’s methods in *Bugiel*’s secure execution environment, which includes implementing the IKE or the SSL/TLS Handshake Protocol to establish a secure communication session. For example, the alleged combinations use *Paaske*’s security engine to perform the cryptographic key exchanges, generate and use keys based on the IKE or SSL handshake protocol, encrypt keys using *Paaske*’s security key, and store relevant secure parameters (such as encrypted secrets/keys, random nonces, DH values) in its secure memory for protection. See *Bugiel* at 221-222, 225-227, 231 (describing various functions performed in its *Bugiel*’s design).

In addition, *Tugenberg* also relates to the same technical area of performing secure communication and key management in a mobile device as *Bugiel*. See generally *Tugenberg* describing secure communication based on e.g., SSL, using a portable communication device, including, e.g., Abstract (“A method for purchasing items over a non-secure communication channel”), 1:15-36 (“Many portable communication devices provide connectivity to networks such as the internet”), 5:47-54 (“Secure Socket Layer (SSL), for example, is a standard technique used in the internet community to establish such a session”); see also *Tugenberg* at 1:40-60, 2:31-4:31, 4:37-68, 5:10-15, 7:8-8:20, Figs. 1 and 3.

Relevant to *Bugiel*’s secure execution environment on SoC, *Tugenberg*’s Figure 1 illustrates a secure execution environment, e.g., host processor 16 and secure memory 20 (together the claimed “security engine”), which are separate from other on-chip system components 15 (e.g., system processor) (including the claimed “processor core”) of the secure processing system 14 (the claimed “system-on-a-chip”). See *Tugenberg* at 5:65-67 (“[T]he secure memory and the host processor are on the same chip.”). On-chip system component 15 reads on the claimed “processor core” because *Tugenberg* explains: “On-chip system components 15 comprise **other system processors**, memory and other functional elements that are part of communication device 10 to allow for wireless or wireline communication of video, data and/or voice.” *Tugenberg* at 2:54-58 (emphasis added); see also *Tugenberg* at 2:31-58, 5:65-67, 8:7-20, Fig. 1. *Tugenberg* also discloses its host processor (part of the security engine) may perform cryptographic functions relevant in *Bugiel* or IKE or SSL/TLS

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

protocols. See *Tugenberg* at 5:47-54 (SSL), 2:3-21 (encryption logic circuitry), 2:59-3:18 (encryption logic circuitry), 4:20-31 (encryption logic circuitry), 6:50-58 (encryption logic circuitry, secure memory), Fig. 1, claims 9 and 17.

In addition, relevant to *Bugiel*'s secure memory, *Tugenberg* discloses its security engine includes secure memory 20, which constitutes a "secure memory" as claimed. *Tugenberg* explains, for example, that "a secure memory embedded in the Host processor is used to secure the user's sensitive information." *Tugenberg* at 4:59-61; see also *id.* at Abstract, 1:7-10, 1:48-54, 2:3-21, 2:31-3:18, 3:34-37, 3:44-4:9, 4:59-61, 5:62-6:2, 6:12-20, 6:50-58, 8:1-20, Fig. 1. A POSITA would have understood these memories are "secure memory accessible only by" host processor 16 and secure memory 20 (together, the claimed "security engine"). *Tugenberg* explains, for example, that "[s]ecure memory 20 includes system monitor 25" that "functions to help ensure the proper operation of secure memory 20. For example, improper attempts to access laser-scribed encryption key 21 or memory 24 are detected by the system monitor 25." *Id.* 3:58-62. *Tugenberg* also describes its apparatus including the secure memory "is equally suitable for the safeguarding and use of any sensitive information." *Id.* at 4:50-53, 1:40-47 ("a method and system that securely stores user authentication information and other sensitive information so that it cannot be easily extracted from the portable communication device, copied to another device, or modified."), 5:62-6:2 ("the sensitive information is never accessible outside of the host processor and the Secure Memory after it is originally loaded . . . the sensitive information is never available outside of this chip, thus protecting the information from any external attempt to intercept it."); see also *id.* 1:15-25, 1:48-55, 2:3-30, 4:59-5:9. This is consistent with *Bugiel*'s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. *Bugiel* at 222-225, 228, 231-234.

In addition, relevant to *Bugiel*'s security key, *Tugenberg*'s Figure 1 shows that secure memory 20 includes a laser-scribed encryption key 21 (the claimed "security key"). See *Tugenberg* at 2:61-63 ("Secure memory 20 also includes laser-scribed encryption key 21 coupled to blocking gates 22."), 6:16-18 (describing "a secure memory that includes the laser-scribed encryption key"), claim 1 (reciting "a secure memory that includes a laser-scribed encryption key"), Figs. 1-3. *Tugenberg* explains "the laser-scribed encryption key is generated by laser-scribing a semiconductor die during fabrication of the secure memory to create a plurality of fixed 'ones' and 'zeroes' which make up the laser-scribed encryption key." *Id.* at claim 7; see also *id.* at claim 8 (reciting "the laser-scribed encryption key is generated burning one-time programmable fuses on a semiconductor die during fabrication of the secure memory to create a plurality of fixed 'ones' and 'zeroes' which make up the laser-scribed encryption key"), 2:16-18 ("[T]he laser-scribed encryption key is generated by laser-scribing a semiconductor die during fabrication."), 3:29-49 ("Laser-scribed encryption key 21 is a randomly generated sequence of 'ones' and 'zeros' suitable for an encryption key. The sequence is laser-scribed during fabrication of a semiconductor die . . ."). *Tugenberg* further explains that "laser-scribed encryption key 21 is comprised of a one-time programmable flash memory element, or alternatively non-volatile memory such as ROM, EEPROM, MRAM (Magnetoresistive RAM), battery backed RAM or DRAM, or other fast logic elements." *Id.* at 3:29-49. Accordingly, *Tugenberg*'s secure memory (secure memory 20) includes a security key (laser-scribed encryption key 21) that was encoded in the secure memory during a manufacturing process of the system-on-a-chip (secure processing

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

system 14). *See also Tugenberg* at Abstract, 2:3-21, 2:59-3:18, 3:58-4:9, 4:20-31, 5:1-9, 6:12-20, 6:40-7:7, 7:17-34, 7:62-67, claims 1-2, 7-8, 10-12, 14-19.

Tugenberg also teaches using laser-scribed encryption key 21 (the claimed “security key”) to encrypt a secret key / common session key. *See Tugenberg* at 4:23-27 (“[E]ach secret key [is] encrypted with laser-scribed encryption key 21 using the secure processing system 14. When needed, communication device decrypts one of the secret keys with encryption logic circuitry 23 using laser-scribed encryption key 21 . . .”), 2:12-15 (“The secure memory uses a laser-scribed encryption key coupled to encryption logic circuitry within the secure memory for encrypting and decrypting sensitive information.”), 2:64-66 (“Encryption logic circuitry 23 preferably implements a symmetric encryption algorithm using laser-scribed encryption key 21.”); *see also Tugenberg* at Abstract, 2:3-21, 2:59-3:18, 3:29-49, 3:58-4:9, 4:20-31, 5:1-9, 6:12-20, 6:40-7:7, 7:17-34, 7:62-67, claims 1-2, 7-8, 10-12, 14-19. *Tugenberg*’s description of using a laser-scribed encryption key that is “unique to each device” and only accessible within the secure memory, to encrypt other keys is consistent with *Bugiel*’s description of using a key that is never available outside TrEE to encrypt the session key and pre-master key. *See Tugenberg* at 2:3-21; *Bugiel* 225-227, 231.

Therefore, a POSITA would have found it obvious to implement *Bugiel*’s method in *Tugenberg*’s SoC apparatus. The combinations based on *Tugenberg* alleged here incorporates *Tugenberg*’s SoC hardware to implement *Bugiel*’s methods in *Bugiel*’s secure execution environment, which includes implementing known protocols, e.g., the IKE or the SSL/TLS Handshake Protocol, etc., to establish a secure communication session. For example, the alleged combinations use *Tugenberg*’s security engine to perform the cryptographic key exchanges, generate and use keys based on the IKE or SSL handshake protocol, encrypt keys using *Tugenberg*’s security key, and store relevant secure parameters (such as encrypted secrets/keys, random nonces, DH values, etc.) in its secure memory for protection. *See Bugiel* at 221-222, 225-227, 231 (describing various functions performed in its *Bugiel*’s design).

In addition, *Aaraj* also relates to the same technical area of performing secure communication and key management in a mobile device as *Bugiel*. *See Aaraj* at 8:15-16 (SSL in secure web browsing between a mobile phone and a web server), 8:3 (Trusted Mobile Platform); *see also Aaraj* at 8:12-13, 8:30-8:31. Relevant to *Bugiel*’s secure execution environment on SoC, *Aaraj* discloses implementing a TPM in a “protected execution domain” in multiprocessor SoC. *See Aaraj* at 8:1-8:3, 8:7, 8:19-8:21, 8:23, 8:26-8:28, 8:30-8:31. Therefore, *Aaraj* teaches the TPM is separate from a processor core in the multiprocessor SoC. *Aaraj* also discloses the TPM may perform cryptographic functions relevant in *Bugiel* or IKE or SSL/TLS protocols. *See Aaraj* at 8:1 (RSA, cryptography), 8:3-8:4 (RSA), 8:6-8:9 (random number generator, RSA, symmetric encryption/decryption, HMAC, SHA-1, SSL, cryptographic functions), 8:7 (encryption/decryption, RSA, HMAC, SHA-1), 8:8 (RSA), 8:9 (SSL, RSA, HMAC, SHA-1, cryptographic functions), 8:15 (SSL, RSA, SHA), 8:31 (SSL).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

In addition, relevant to *Bugiel*'s secure memory, *Aaraj* discloses TPM provides “capabilities for secure storage.” See *Aaraj* at 8:2; see also *Aaraj* at 8:3, 8:5-8:7, 8:15, 8:30-8:31. *Aaraj* further discloses “[p]rotected storage is accessible only within the TPM, and is shielded from physical attacks.” *Aaraj* at 8:5. This is consistent with *Bugiel*'s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. *Bugiel* at 222-225, 228, 231-234.

In addition, relevant to *Bugiel*'s security key, *Aaraj* discloses “the endorsement key (EK), is generated by the vendor or manufacturer, and stored in the nonvolatile memory within the TPM. EK should never be disclosed and is only used for . . . creating the storage root key (SRK). . . . The SRK is the root of the hierarchy of TPM keys, and is used for encrypting other migratable keys that are stored outside the TPM.” *Aaraj* at 8:6. *Aaraj*'s description of using EK that should never be disclosed to encrypt other keys is consistent with *Bugiel*'s description of using a key that is never available outside TrEE to encrypt the session key and pre-master key. See *Tugenberg* at 2:3-21; *Bugiel* 225-227, 231.

Therefore, a POSITA would have found it obvious to implement *Bugiel*'s method in *Aaraj*'s SoC apparatus. The combinations based on *Aaraj* alleged here incorporate *Aaraj*'s SoC hardware to implement *Bugiel*'s methods in *Bugiel*'s secure execution environment, which includes implementing the IKE or the SSL/TLS Handshake Protocol to establish a secure communication session. For example, the alleged combinations use *Aaraj*'s security engine to perform the cryptographic key exchanges, generate and use keys based on known protocols, such as the IKE or SSL handshake protocol, etc., encrypt keys using *Aaraj*'s security key, and store relevant secure parameters (such as encrypted secrets/keys, random nonces, DH values, etc.) in its secure memory for protection. See *Bugiel* at 221-222, 225-227, 231 (describing various functions performed in its *Bugiel*'s design).

In addition, *Gibod* also relates to the same technical area of performing secure communication and key management in a mobile device as *Bugiel*. See *Gibod* ¶ [0066] (“cellular telephone, personal digital assistance, portable audio or video player, television set-top box, kiosk, gaming system, other embedded device, or the like. . . . communicating with other systems via a network 516 such as, for example, the Internet, a local area network, a virtual private network, and the like using one or more communication technologies (e.g., wireless, Ethernet, and/or the like)”). Relevant to *Bugiel*'s secure execution environment on SoC, *Gibod* discloses a secure processing unit (“SPU”) 518 is separate from processing unit 502 on a SoC in Fig. 5. See *Gibod* ¶¶ [0065]-[0067]; see also *Gibod* at Abstract (describing a secure processing domain separate from an open processing domain), ¶¶ [0005]-[0009], ¶¶ [0018]-[0020], ¶¶ [0037]-[0039], ¶¶ [0041], ¶ [0047], ¶¶ [0069]-[0070], Figs. 1 (Secure Processor separate from Media Processing, CPU, GPU and other Operating System) and 5, claims 1, 3-5, 14, 16, 18, and 19. *Gibod* also discloses its security engine may perform cryptographic functions relevant in *Bugiel*, IKE or SSL/TLS protocols. See *Gibod* at ¶¶ [0008]-[0009] (encryption/decryption), ¶ [0019] (encryption/decryption), [0033]-[0037] (RSA, symmetric encryption/decryption, signing, hashing), ¶ [0052] (random number generator), ¶ [0055] (encryption/decryption), ¶ [0067] (key management), ¶ [0071] (TLS, hash, symmetric encryption, public key exchange), ¶¶ [0070]-[0071], Fig. 1 (RSA, AES, SHA), claim 19 (a secure processor including integrated support for symmetric and asymmetric cryptographic algorithms).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

In addition, relevant to *Bugiel*'s secure memory, *Gibod* teaches this claim feature by disclosing “protected memory space of the security processor.” *Gibod* ¶ [0019]; *see also* *Gibod* ¶¶ [0004]-[0005] (“a secure processing domain may be established where secret information is securely processed and protected from exposure to applications executing outside the secure processing domain”), ¶ [0067] (“the SPU 518 may include internal memory”), ¶ [0023] (“hardware security element (e.g., One Time Programmable [‘OTP’] memory”), ¶¶ [0005]-[0007], ¶ [0016], ¶¶ [0025]-[0026], ¶ [0046], ¶ [0039], ¶ [0041], ¶ [0049], ¶ [0053], ¶ [0058], ¶ [0067], ¶¶ [0069]-[0070], claim 1. This is consistent with *Bugiel*'s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. *Bugiel* at 222-225, 228, 231-234.

In addition, relevant to *Bugiel*'s security key, *Gibod* discloses a device key/Master key that is “built-in” permanently in the device. *Gibod* ¶ [0051]; *see also* *Gibod* ¶ [0021], ¶ [0023], ¶ [0037], Fig. 1. *Gibod* also discloses the Master key is stored in secure memory, e.g., “secure storage.” *Gibod* ¶ [0026], claim 14.

Gibod further discloses using the Master key to encrypt/decrypt other keys. *See* *Gibod* ¶¶ [0009]-[0010] (“[t]o export or persist secret information, the secret information may first be encrypted using a persistent (e.g., permanent) encryption key.”), ¶ [0019] (“... secret information (e.g., cryptographic keys) . . .”), ¶ [0004] (“... information (e.g., cryptographic keys, . . .) . . . designed to prevent unauthorized access and/or use to secure and/or proprietary content may also need to be kept secret . . .”), ¶ [0008] (“performing encryption/decryption operations utilizing the secret information, . . . , encrypting secret fields within a larger piece of data utilizing the secret information”), ¶ [0032], ¶ [0004], ¶¶ [0021]-[0024], ¶¶ [0026]-[0032], ¶ [0046], ¶¶ [0050]-[0051], ¶ [0053], ¶¶ [0069]-[0070], claims 11-14. *Gibod*'s description of using a master key that is securely embedded in a hardware security element to encrypt other keys is consistent with *Bugiel*'s description of using a key that is never available outside TrEE to encrypt the session key and pre-master key. *See* *Bugiel* 225-227, 231.

Therefore, a POSITA would have found it obvious to implement *Bugiel*'s method in *Gibod*'s SoC apparatus. The combinations based on *Gibod* alleged here incorporate *Gibod*'s SoC hardware to implement *Bugiel*'s methods in secure execution environment, which includes implementing the IKE or the SSL/TLS Handshake Protocol to establish a secure communication session. For example, the alleged combinations use *Gibod*'s security engine to perform the cryptographic key exchanges, generate and use keys based on known protocols, such as the IKE or SSL handshake protocol, etc., encrypt keys using *Gibod*'s security key, and store relevant secure parameters (such as encrypted secrets/keys, random nonces, DH values, etc.) in its secure memory for protection. *See* *Bugiel* at 221-222, 225-227, 231 (describing various functions performed in its *Bugiel*'s design).

In addition, *Mujtaba* also relates to the same technical area of performing secure communication and key management in a mobile device as *Bugiel*. *See* *Mujtaba* ¶¶ [0025]-[0026] (“device 100 may represent a Smartphone such as an iPhoneTM from Apple Inc. of Cupertino, Calif. Alternatively, device 100 may represent a tablet PC such as an iPadTM from Apple Inc. . . . Wireless processor 101 may be any kind of wireless processors, such as, for example, a cellular processor, a Wi-Fi processor, a Bluetooth processor, etc.”),

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

¶ [0051] (“a system configuration for securely downloading data from a remote facility over a network”), ¶ [0052] (“secure connection”); *see also Mujtaba* ¶¶ [0003]-[0004], ¶¶ [0058]-[0062], ¶¶ [0051]-[0053], ¶ [0059], Abstract. Relevant to *Bugiel*’s secure execution environment on SoC, *Mujtaba* illustrates in Figure 9, “a processing system 911, which may be one or more microprocessors, or which may be a system on a chip of integrated circuit.” *Mujtaba* ¶ [0058]. The processing system 911 may comprise processors such as wireless communication processor 101, which is separate from a processor core application processor (e.g., general purpose processor) 102 illustrated in Figure 1. *See also Mujtaba* ¶¶ [0026]-[0034], ¶¶ [0063]-[0068]. *Mujtaba* also discloses its wireless communication processor may perform the cryptographic functions relevant in *Bugiel*, IKE or SSL/TLS protocols. *See Mujtaba* at ¶ [0006] (session key generation, encryption/decryption, key management), ¶¶ [0042]-[0044] (cryptographic unit for encryption), ¶¶ [0052]-[0053] (session key generation, encryption/decryption, key management), ¶ [0039] (encryption), ¶¶ [0047]-[0050] (cryptographic unit, encryption), ¶ [0055]-[0057] (encryption/decryption), ¶¶ [0066]-[0068], Fig. 4.

In addition, relevant to *Bugiel*’s secure memory, *Mujtaba* teaches this claim feature by illustrating secure memory accessible only by the security engine wireless communication processor 101, including e.g., public key storage 119, secure ROM 112, internal RAM 113, storage key 114, RAM 103, FS Driver 153 etc. in Figures 1 and 4. *See also Mujtaba* ¶ [0028], ¶ [0031], ¶ [0033], ¶ [0037], ¶ [0044], ¶¶ [0064]-[0068], Figure 4. This is consistent with *Bugiel*’s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. *See Bugiel* at 222-225, 228, 231-234.

In addition, relevant to *Bugiel*’s security key, *Mujtaba* discloses “Storage key 114 and/or the UID (not shown) may be hardwired in the hardware (e.g., burnt fuses) during the manufacturing of wireless processor 101.” *Mujtaba* ¶ [0031], ¶ [0033], ¶ [0037], ¶¶ [0066]-[0068], Figs. 1 and 4. Figures 1 and 4 illustrate the storage key 114 is inside secure memory accessible only by the security engine wireless processor 101. *Mujtaba* further discloses the storage key or another key derived from the UID is used to encrypt a generated session key. *See Mujtaba* ¶ [0006] (“A recovery blob is generated having embedded therein the session key encrypted by the storage key. . . . Thereafter, the recovery blob . . . are sent to an authorization server.”), ¶¶ [0052]-[0053] (“a recovery blob is generated by encrypting session key 705 using storage key 114”); *see also* ¶ [0031], ¶¶ [0042]-[0044], ¶ [0055], ¶ [0057], ¶¶ [0066]-[0068], claims 10, 18, 23, Figs. 1, 4, 7-8. Under Daedalus’s apparent view, *Mujtaba*’s description of using a storage key or a key derived from UID that is “locally maintained” and “stored in the secure ROM” to encrypt other keys is consistent with *Bugiel*’s description of using a key that is never available outside TrEE to encrypt the session key and pre-master key. *Mujtaba* ¶ [0049], ¶ [0023], ¶ [0037]; *see Bugiel* 225-227, 231.

Therefore, a POSITA would have found it obvious to implement *Bugiel*’s method in *Mujtaba*’s SoC apparatus. The combinations based on *Mujtaba* alleged here incorporates *Mujtaba*’s SoC hardware to implement *Bugiel*’s methods in secure execution environment, which includes implementing known protocols, such as the IKE or the SSL/TLS Handshake Protocol, etc. to establish a secure communication session. For example, the alleged combinations use *Mujtaba*’s security engine to perform the cryptographic key exchanges, generate and use keys based on the IKE or SSL handshake protocol, encrypt keys using *Mujtaba*’s security key, and store relevant secure

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

parameters (such as encrypted secrets/keys, random nonces, DH values, etc.) in its secure memory for protection. *See Bugiel* at 221-222, 225-227, 231 (describing various functions performed in its *Bugiel*'s design).

In addition, *Buer754* also relates to the same technical area of performing secure communication and key management in a mobile device as *Bugiel*. *See Buer754* at Title (“Key Management System and Method”), 1:26-2:8 (IPsec IKE and SSL); *see also Buer754* at 2:24-39, 5:24-29, 6:13-31, 11:46-51, 12:40-44, 13:8-13, 14:60-15:3, 15:17-16:35. Relevant to *Bugiel*'s secure execution environment on SoC, Fig. 3 of *Buer754* discloses security module 320 and cryptographic accelerator 326 (collectively the security engine) are separate from host processor. *Buer754* at 6:13-23; *see also Buer754* at 4:61-5:6, 6:13-35, 10:14-28, 11:62-12:3, 17:48-62, Figs. 1-3, 10. *Buer754* also discloses the security engine may perform cryptographic functions relevant in *Bugiel*, IKE or SSL/TLS protocols. *See Buer754* at 1:26-2:13 (SSL, IPsec IKE, RSA key exchange, encryption/decryption), 6:13-31 (SSL, public key exchange), 8:63-9:4 (IKE), 13:14-15:11 (Diffie-Hellman, RSA, HMAC), 15:17-16:35 (IPsec, HMAC, SHA1, SSL/TLS, RSA, DSA, Diffie-Hellman, 3DES, AES); *see also Buer754* at 5:24-29, 9:16-21.

In addition, relevant to *Bugiel*'s secure memory, Figure 3 of *Buer754* illustrates secure memory accessible only by security module 320 and cryptographic 326 (collectively the security engine), including EEPROM 336, KEK 332, private keys 330, etc. In addition, Figure 4 illustrates secure memory accessible only by the cryptographic accelerator 420 (part of the security engine), including e.g., memories 432A and 432B, EEPROM 434, etc. Figure 5 illustrates secure memory accessible only by cryptographic accelerator 520 (part of the security engine), including e.g., RD_EEPROM, WR_EEPROM, data storages 532, 534 and 536 etc. *See also Buer754* at 2:17-60, 4:45-60, 5:52-62, 7:59-64, 8:51-56, 10:29-41, 12:45-61, 13:66-14:6, 15:17-32, 15:46-60, 16:63-67, 17:1-62, Figs. 1-5 and 10, claims 1, 9-11, 13. *Buer754* also describes protecting keys in the secure memory. *See Buer754* at 3:49-57 (“the keys are stored in a ‘protected portion’ of the system S”), 5:52-62 (“unencrypted security associations never leave the protected cryptographic accelerator 236”), 15:4-11 (“The shared secret must be protected within the cryptographic boundary”); *see also Buer754* at 10:29-41, 12:24-39, 14:35-42. This is consistent with *Bugiel*'s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. *See Bugiel* at 222-225, 228, 231-234.

In addition, relevant to *Bugiel*'s security key, *Buer754* discloses “the initial KEK 130 may be programmed into the non-volatile memory 134 when the system is manufactured.” *Buer754* at 4:45-60; *see also Buer754* at 3:54-56, 5:52-62, 8:1-8, 10:34-37, 13:66-14:10, 17:5-26, 17:48-62.

Buer754 also describes using the KEK to encrypt other keys by disclosing “these session keys are encrypted using KEK 238 and stored in a database 240 as security associations.” *Buer754*, 5:43-51; 6:36-43 (“To protect these keys, the security module 320 encrypts them with a KEK 332 before they are sent to the cryptographic accelerator 326”), 2:50-60 (“using a single KEK to encrypt all the keys associated with the network data”), Fig. 4 (encrypted packet & security association sent to cryptographic accelerator 420), Fig. 6 (step 610: encrypt and store session keys), Fig. 11 (steps 1118-1120 : encrypt and store private keys using KEK), Fig. 12 (steps 1210-1212:

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

encrypt session keys using KEK_A and store security associations), 4:45-60 (“the initial KEK 130 may be programmed into the non-volatile memory 134 when the system is manufactured.”); *see also Buer754* at 1:42-49, 2:9-12, 2:25-60, 3:54-56, 12:47-52, 4:14-26, 5:52-6:12, 6:53-64, 6:65-7:23, 7:35-41, 8:1-8, 8:35-50, 9:5-12, 9:22-44, 10:23-27, 10:34-37, 10:62-11:3, 11:17-31, 11:62-12:3, 13:66-14:10, 15:17-33, 16:8-15, 17:5-26, 17:48-62; Figs. 3, 5, and 13, claims 1-3, 6-9, 12, 14, 16. *Buer754*’s description of using a KEK to encrypt other keys is consistent with *Bugiel*’s description of using a key that is never available outside TrEE to encrypt the session key and pre-master key. *See Bugiel* 225-227, 231.

Therefore, a POSITA would have found it obvious to implement *Bugiel*’s method in *Buer754*’s SoC apparatus. The combinations based on *Buer754* alleged here incorporate *Buer754*’s SoC hardware to implement *Bugiel*’s methods in secure execution environment, which includes implementing known protocols, such as the IKE or the SSL/TLS Handshake Protocol, etc., to establish a secure communication session. For example, the alleged combinations use *Buer754*’s security engine to perform the cryptographic key exchanges, generate and use keys based on the IKE or SSL handshake protocol, encrypt keys using *Buer754*’s security key, and store relevant secure parameters (such as encrypted secrets/keys, random nonces, DH values, etc.) in its secure memory for protection. *See Bugiel* at 221-222, 225-227, 231 (describing various functions performed in its *Bugiel*’s design).

Multiple rationales support such an application of one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754* to *Bugiel*’s teachings:

- A combination of prior art elements (security engine in a system-on-a-chip apparatus disclosed in *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754* and *Bugiel*’s secure communication channel) according to known methods (*Bugiel*’s teaching of establish secure communication) to yield predictable results (establishing secure communication between a client mobile device and a server);
- Using a known technique (implementing *Bugiel*’s technique for secure communications using mobile devices) to improve similar devices (*Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*’s flexible security hardware for mobile devices) in the same way (by using *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*’s hardware to implement *Bugiel*’s technique and thereby establish secure communications between a client mobile device and a server);
- Applying a known technique (implementing *Bugiel*’s technique for secure communications using mobile devices) to a known device (*Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*’s flexible security hardware for mobile devices) ready for improvement (by using *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*’s hardware to implement *Bugiel*’s technique and thereby establish secure communications between a client mobile device and a server); and

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

- A teaching, suggestion, or motivation in the prior art (e.g., *Bugiel*'s teaching to use secure hardware for mobile devices that provides a trusted execution environment to protect the execution of critical code from tampering, and *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s disclosure of such hardware) that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings (as described above) to arrive at the claimed invention.

A POSITA thus would have found it obvious to use one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba* or *Buer754*'s secure hardware to implement *Bugiel*'s secure communication technique in secure execution environment, which includes using the known protocols described in *Bugiel*, e.g., *TLS-1.2*. See, e.g., *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 416-21 (2007).

Exemplary Reasons to Implement SSL/TLS Handshake Protocols Based on *Bugiel*

A prior-art reference that expressly incorporates by reference material from another source may rely on the incorporated material as if that material is within the “four corners” of the reference. “To incorporate matter by reference, a host document must contain language ‘clearly identifying the subject matter which is incorporated and where it is to be found.’” *Callaway Golf Co. v. Acushnet Co.*, 576 F.3d 1331, 1346 (Fed. Cir. 2009) (quoting *In re De Seversky*, 474 F.2d 671, 674 (CCPA 1973)). Here, *Bugiel* explicitly references *TLS-1.2* regarding how to implement SSL/TLS handshake protocol messages: “All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16].” See *Bugiel* at 225 (citing *TLS-1.2* as document [16]), 235-236. Thus, all teachings of *TLS-1.2* regarding “standard messages of SSL/TLS handshake” may be considered as though they appear in *Bugiel* itself.

Additionally, a POSITA also would have found it obvious to combine the teachings of one of the SSL/TLS protocols (*SSL-3.0*, *TLS-1.0*, *TLS-1.1*, or *TLS-1.2*) with those in *Bugiel* and one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*.

For example, a POSITA would have found it obvious to implement one of the SSL/TLS handshake protocols while using one of *Bugiel*, *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s security hardware to implement *Bugiel*'s secure communications technique in secure execution environment. *Bugiel* explains: “All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification.” *Bugiel* at 225. A POSITA thus would have found it obvious to reference the teachings in a SSL/TLS protocol, e.g., *TLS-1.2* for how to send and receive SSL/TLS messages (including how to generate a random number and include the random number in a ClientHello message to initiate secure communication with a remote server) as described in *Bugiel*.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

In addition, SSL/TLS was one of the most popular secure communication protocols at the relevant time. As one textbook described at the time, “Support for SSL is now built in to nearly all browsers and Web servers.” *SSL & TLS Essentials* at 6.³ As that book further explains, the Internet Engineering Task Force (IETF) “renamed SSL to Transport Layer Security (TLS) Despite the change of names, TLS is nothing more than a new version of SSL.” *Id.* '838 patent also admits “[o]ne commonly used cryptographic communication protocol is the Secure Socket Layer (SSL) protocol. The SSL protocol, and the related Transport Layer Security (TLS) protocol, is used for many types of secure communication sessions” in its Background section. '838 patent at 1:8-32. In the prior Samsung IPR against the '838 patent, Samsung’s expert Dr. Mark E. Crovella described the history and popularity of the SSL/TLS protocols at the earliest possible priority date of the '838 patent in his expert declaration. *See Crovella*⁴ ¶¶ 61-70.

Each of *SSL-3.0*, *TLS-1.0*, *TLS-1.1*, and *TLS-1.2* protocols is a revision of the previous version of SSL/TLS protocols. *See TLS-1.0* at 5 (Section “3. Goals of this document: This document and the TLS protocol itself are based on the SSL 3.0 Protocol Specification as published by Netscape. The differences between this protocol and SSL 3.0 are not dramatic, but they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate (although TLS 1.0 does incorporate a mechanism by which a TLS implementation can back down to SSL 3.0).”); *TLS-1.1* at 5 (Section “1.1. Differences from TLS 1.0: This document is a revision of the TLS 1.0 [TLS1.0] protocol, and contains some small security improvements, clarifications, and editorial improvements.”); *TLS-1.2* at 5-6 (Section “1.2 Major Differences from TLS 1.1: This document is a revision of the TLS 1.1 [TLS1.1] protocol which contains improved flexibility, particularly for negotiation of cryptographic algorithms.”).⁵ In addition, each of *SSL-3.0*, *TLS-1.0*, *TLS-1.1*, and *TLS-1.2* protocols is designed to be backward-compatible with previous versions of SSL/TLS protocols. *See TLS-1.2* at 87-88 (Compatibility with *TLS-1.0*, *TLS-1.1*, and *SSL-3.0*); 88-90 (Compatibility with SSL version 2.0); *TLS-1.1* at 71 (Compatibility with *TLS-1.0* and *SSL-3.0*); *TLS-1.0* at 66 (Compatibility with *SSL-3.0*). *TLS-1.2* states that “TLS versions 1.0, 1.1, and 1.2, and SSL 3.0 are very similar . . . ; thus, supporting all of them is relatively easy.” *TLS-1.2* at 87; *see also TLS-1.1* at 71 (“TLS versions 1.1 and 1.0, and SSL 3.0 are very similar; thus, supporting both is easy.”); *TLS-1.0* at 66 (“TLS version 1.0 and SSL 3.0 are very similar; thus, supporting both is easy.”). Therefore, a POSITA would have been motivated to reference one of *SSL-3.0*, *TLS-1.0*, *TLS-1.1*, or *TLS-1.2* for implementation details regarding how to implement the SSL/TLS Handshake Protocol.

³ Stephen Thomas, *SSL and TLS Essentials: Securing the Web* (2000).

⁴ Mark E. Crovella, Declaration Of Dr. Mark E. Crovella In Support Of Petition For Inter Partes Review Of U.S. Patent No. 9,887,838 (“*Crovella*”), IPR2023-00727, Exhibit 1002 (P.T.A.B March 20, 2023).

⁵ The cited page numbers are the page numbers # in “[Page #]” printed on each of the *SSL-3.0*, *TLS-1.0*, *TLS-1.1*, and *TLS-1.2* protocols instead of the pdf numbers.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

Given the express teachings in *Bugiel* and the ubiquity of the SSL/TLS communication protocol, a POSITA would have been motivated to review the relevant SSL/TLS protocol for more details regarding how to implement the TLS Handshake Protocol for generating cryptographic keys based on cryptographic key exchange.

The grounds alleged here incorporates one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s SoC apparatus to implement *Bugiel*'s secure communications technique, including the SSL/TLS protocol's teachings that *Bugiel* incorporates by reference. For example, a POSITA would have found it obvious to use *Chou*'s HSM 20 and SPU-M 39 to generate and use keys based on the SSL protocol when implementing *Bugiel*'s secure communications technique, which involves using SSL/TLS protocol, e.g., *TLS-1.2* handshake protocol messages. See *Chou* ¶¶ [0042]-[0055], ¶ [0018], ¶ [0066] (describing the capabilities of its hardware components); *Bugiel* at 225 ("All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding [*TLS-1.2*] specification."); *TLS-1.2*, at 91 ("The TLS protocol is designed to establish a secure connection between a client and a server communicating over an insecure channel"), 6 ("Cryptographic security: TLS should be used to establish a secure connection between parties"), 33-37 ("The cryptographic parameters of the session state are produced by the TLS Handshake Protocol, which operates on top of the TLS record layer. When a TLS client and server first start communicating, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public-key encryption techniques to generate shared secrets."), 96 ("For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure"). *SSL-3.0*, *TLS-1.0*, and *TLS-1.1* provide similar disclosures as *TLS-1.2*. See *TLS-1.1*, at 5-6, 31-34, 74, 80; *TLS-1.0*, at 4-5, 29-32, 69, 73; *SSL-3.0* at 4-5, 18-20, 55, 59. In addition to *Chou*, the same logic applies to combinations based on other references.

Further, each of the TLS/SSL protocol had been in practice for more than three years at the time the '838 patent was filed. Accordingly, the TLS handshake process described in each of the SSL/TLS protocols, including *TLS-1.2* would have been mature enough for a POSITA to implement in one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s system-on-a-chip architecture with *Bugiel*'s teaching.

Multiple rationales support such application of one of the SSL/TLS protocols (*SSL-3.0*, *TLS-1.0*, *TLS-1.1*, and *TLS-1.2*)'s teachings to *Bugiel* alone or the combination of *Bugiel* and one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754* described above:

- A combination of prior art elements (security engine in a system-on-a-chip apparatus disclosed in one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754* and the SSL/TLS handshake messages described in the SSL/TLS protocols (*SSL-3.0*, *TLS-1.0*, *TLS-1.1*, and *TLS-1.2*) and relevant to *Bugiel* to yield predictable results (establishing secure communication between a client and a server);

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

- Using a known technique (implementing *Bugiel*'s technique for secure communications on mobile devices, which uses the SSL/TLS protocols, e.g., *TLS-1.2* handshake protocol messages) to improve similar devices (one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s flexible security hardware for mobile devices) in the same way (by using one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s hardware to implement *Bugiel*'s technique, which uses the SSL/TLS handshake protocol messages, and thereby establish secure communications between a client mobile device and a server);
- Applying a known technique (implementing *Bugiel*'s technique for secure communications on mobile devices, which uses the SSL/TLS handshake protocol messages, e.g., *TLS-1.2*) to a known device (one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s flexible security hardware for mobile devices) ready for improvement (by using one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s hardware to implement *Bugiel*'s technique, which uses the SSL/TLS, e.g., *TLS-1.2* handshake protocol messages, and thereby establish secure communications between a client mobile device and a server); and
- A teaching, suggestion, or motivation in the prior art (e.g., *Bugiel*'s teaching to use the SSL/TLS protocols, e.g., *TLS-1.2* handshake protocol messages and secure hardware for mobile devices with a trusted execution environment to protect the execution of critical code from tampering, one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s disclosure of the requisite hardware, and the SSL/TLS protocols, e.g., *TLS-1.2*'s teaching of the requisite handshake protocol messages) that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings (as described above) to arrive at the claimed invention.

A POSITA thus would have found it obvious to use one of *Bugiel*, *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s secure hardware to implement *Bugiel*'s secure communication technique in secure execution environment, which includes using the handshake protocol messages described in one of the SSL/TLS protocols, e.g., *TLS-1.2*. See, e.g., *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 416-21 (2007).

Exemplary Reasons to Implement IPsec IKE Protocols Based on *Bugiel*

IKE protocol is used in IPsec to establish secure communication channel between a client device and server. See *RFC-5996* at 1, 5-9; *RFC-4306* at 1, 3-7. *Bugiel* describes its goal is to establish a "secure channel between the user and the web server." *Bugiel* at 221; see also *Bugiel* at 223, 234-236. Further, both IPsec IKE and SSL/TLS protocols are secure communication protocols that establish session key based on cryptographic key exchange such as Diffie-Hellman. See *Tkacik* ¶ [0080] ("keys may be generated using any of a number of protocols known to those skilled in the art, including IKE, DH, RSA key gen., SSL, random key, and the like"); *Buer754* at 16:26-35

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

(describing both IPsec and SSL as secure communication protocol), 1:27-42; *see infra* [1b]-[1g], [2], [5]-[8]. Therefore, a POSITA would have found it obvious to establish the secure channel using one of the IKE protocols (*RFC-5996* or *RFC-4306*).

Multiple rationales support such application of one of the IKE protocols (*RFC-5996* or *RFC-4306*)'s teachings to *Bugiel* alone or *Bugiel* in view of one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*:

- A combination of prior art elements (security engine in a system-on-a-chip apparatus disclosed in one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754* and the key exchange process described in one of the IKE protocols (*RFC-5996* or *RFC-4306*) and relevant to *Bugiel* to yield predictable results (establishing secure communication between a client and a server);
- Using a known technique (implementing *Bugiel*'s technique for secure communications on mobile devices, which uses one of the IKE protocols (*RFC-5996* or *RFC-4306*)) to improve similar devices (one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s flexible security hardware for mobile devices) in the same way (by using one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s hardware to implement *Bugiel*'s technique and thereby establish secure communications between a client mobile device and a server);
- Applying a known technique (implementing *Bugiel*'s technique for secure communications on mobile devices, which uses one of the IKE protocols (*RFC-5996* or *RFC-4306*)) to a known device (one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s flexible security hardware for mobile devices) ready for improvement (by using one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s hardware to implement *Bugiel*'s technique and thereby establish secure communications between a client mobile device and a server); and
- A teaching, suggestion, or motivation in the prior art (e.g., *Bugiel*'s teaching to establish secure channel and secure hardware for mobile devices with a trusted execution environment to protect the execution of critical code from tampering, one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*'s disclosure of the requisite hardware, and one of the IKE protocols (*RFC-5996* or *RFC-4306*)'s teaching of establishing secure channel through key exchange) that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings (as described above) to arrive at the claimed invention.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

Therefore, a POSITA would have found it obvious to implement one of the IKE protocols (*RFC-5996*, or *RFC-4306*)’s teachings to *Bugiel* alone or *Bugiel* in view of one of *Chou*, *Tkacik*, *Paaske*, *Tugenberg*, *Aaraj*, *Gibod*, *Mujtaba*, or *Buer754*. See, e.g., *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 416-21 (2007).

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
Claim 1	
[1pre] A system-on-a-chip apparatus comprising:	To the extent the preamble is limiting, the grounds alleged at the beginning of this chart disclose or suggest “[a] system-on-a-chip apparatus.” <i>Bugiel</i> expressly teaches to use secure hardware for mobile devices. See <i>Bugiel</i> at 219-225, 227-236. More specifically, <i>Bugiel</i> teaches to implement its design using any “secure hardware for mobile devices” that provides “a Trusted Execution Environment (TrEE) to protect the execution of critical code from tampering.” <i>Id.</i> at 220; see also <i>id.</i> at 234 (“Our solution does not rely on a secure operating system, but exploits hardware security features, in particular Trusted Execution Environment (TrEE) available on many modern mobile phones.”). In particular, <i>Bugiel</i> describes implementing its method on a Nokia N900 with M-shield technology. See <i>id.</i> at 234, 219, 230. The Nokia N900 was release in November, 2009 and was equipped with OMAP 3430, which is a system-on-a-chip with M-shield technology.⁶ The diagram, titled "OMAP3430 system diagram", illustrates the internal architecture of the OMAP3430 system-on-a-chip. The central chip is connected to various external components. On the left, it interfaces with a "Voice" module (containing a "Voice Codec" and "Voice Processor"), a "Data" module (containing a "Data Codec" and "Data Processor"), and a "Camera" module (containing a "Camera Sensor" and "Camera Processor"). The chip also connects to a "Display" module (containing a "Display Controller" and "Display Processor") and a "Power" module (containing a "Power Manager" and "Power Processor"). Other connections include "USB", "SPI", "I2C", "UART", "GPIO", "JTAG", "EMMC", "NAND", "SDIO", "PCI", "PCIe", "SATA", "FireWire", "Thunderbolt", "Bluetooth", "Wi-Fi", "ZigBee", "4G LTE", "5G", "6G", "7G", "8G", "9G", "10G", "11G", "12G", "13G", "14G", "15G", "16G", "17G", "18G", "19G", "20G", "21G", "22G", "23G", "24G", "25G", "26G", "27G", "28G", "29G", "30G", "31G", "32G", "33G", "34G", "35G", "36G", "37G", "38G", "39G", "40G", "41G", "42G", "43G", "44G", "45G", "46G", "47G", "48G", "49G", "50G", "51G", "52G", "53G", "54G", "55G", "56G", "57G", "58G", "59G", "60G", "61G", "62G", "63G", "64G", "65G", "66G", "67G", "68G", "69G", "70G", "71G", "72G", "73G", "74G", "75G", "76G", "77G", "78G", "79G", "80G", "81G", "82G", "83G", "84G", "85G", "86G", "87G", "88G", "89G", "90G", "91G", "92G", "93G", "94G", "95G", "96G", "97G", "98G", "99G", "100G".

⁶ See https://en.wikipedia.org/wiki/Nokia_N900, https://www.ti.com/pdfs/wtbu/ti_omap3430.pdf.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>Chou</i> discloses “a single integrated circuit (IC) chip 91 that incorporates a complete system on the IC chip (System-On-Chip or SOC).” <i>Chou</i> ¶ [0065]; <i>see also Chou</i> ¶ [0006], ¶¶ [0017]-[0018], ¶ [0027], ¶ [0036], ¶ [0040], ¶¶ [0042]-[0055], ¶ [0063], ¶ [0066], ¶¶ [0068]-[0069], Figs. 1, 3-5. The System-On-Chip 91 maps to the claimed “system-on-a-chip apparatus.” FIG. 4 FIG. 5 <i>Chou</i> at Figs. 4-5 (annotated). <i>Chou</i>’s description that “a single integrated circuit (IC) chip 91 that incorporates a complete system on the IC chip (System-On-Chip or SOC)” is consistent with the ’838 patent’s description that “SOC devices are integrated circuits that incorporate various components, in addition to the processing core, of electronic systems on a single die. For example, an SOC may include a processor core, memory controller, video components, audio components, and/or communication components on a single chip.” ’838 patent at 1:27-32. Therefore, <i>Chou</i>’s System-On-Chip 91 is a “system-on-a-chip apparatus.” In addition, <i>Chou</i> explains that “[t]he particular implementation of SOC 91 in FIG. 4 below is a wireless device 90 that is used to transmit and receive wireless communication.” <i>Chou</i> ¶ [0066].

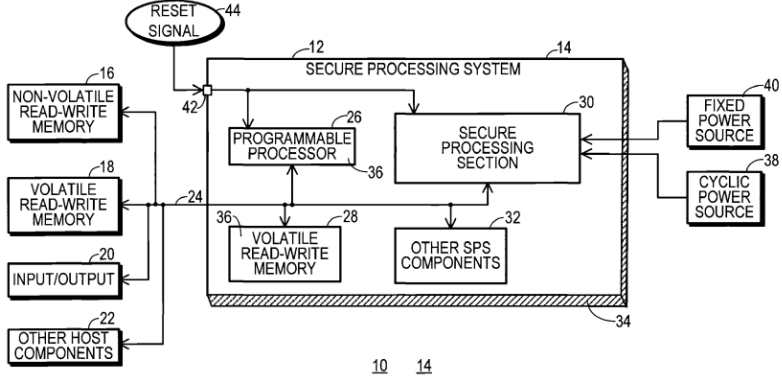
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	FIG. 4 FIG. 5 <i>Chou</i> at Figs. 4 and 5. To the extent that Daedalus argues a device incorporating the system-on-a-chip constitutes the claimed “system-on-a-chip apparatus,” the “wireless device 90” illustrated in <i>Chou</i>’s Figure 4, which incorporates SOC 91, also constitutes the claimed “system-on-a-chip apparatus” as Figure 4 shows above. In addition, mobile phone 102, which includes SOC 91 as Figure 5 shows above, is also the “system-on-a-chip apparatus” under such view. For convenience of discussion, this claim chart maps SOC 91 as the claimed “system-on-a-chip” in claim features [1a-1], [1a-3], and [1g], whereas this chart only shows how to map wireless device 90 (or mobile phone 102) as the claimed “system-on-a-chip apparatus” in [1pre]. Other references also disclose a system-on-a-chip apparatus. For example, <i>Tkacik</i> discloses a secure processing system 12.

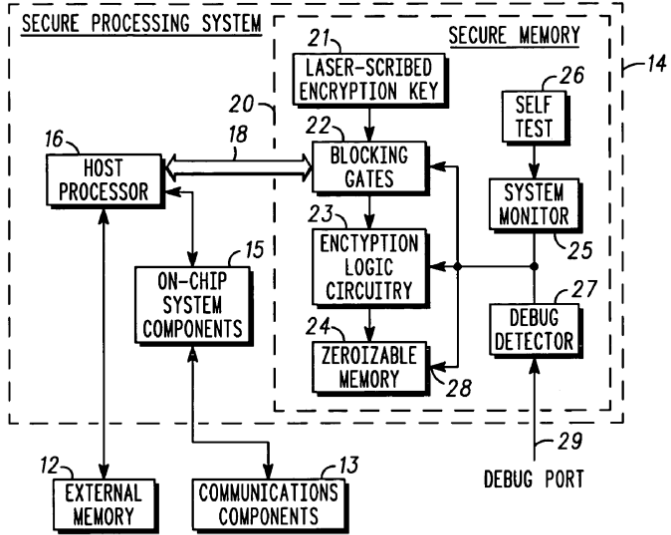
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	FIG. 1 <i>Tkacik</i>, Fig. 1 <i>Tkacik</i> explains that its secure processing system 12 includes “processor 26, memory 28, secure processing section 30, and other SPS components 32,” which “are all formed together on a common semiconductor substrate and are packaged together and reside within a single integrated circuit 34.” <i>Tkacik</i> ¶ [0025], Fig. 1; <i>see also Tkacik</i> at Abstract, ¶ [0020], ¶¶ [0001]-[0002], ¶¶ [0011]-[0012], ¶¶ [0015]-[0018], ¶¶ [0022]-[0023], ¶ [0026], ¶¶ [0029]-[0034], ¶ [0037], ¶¶ [0039]-[0040], ¶ [0046], ¶ [0058], ¶ [0064], ¶ [0066], ¶ [0073], ¶ [0079], ¶ [0089], Figures 1 and 2. The description that all the components of secure processing system 12 “are all formed together on a common semiconductor substrate and are packaged together and reside within a single integrated circuit 34” is consistent with the description in the specification of the '838 patent that “SOC devices are integrated circuits that incorporate various components, in addition to the processing core, of electronic systems on a single die. For example, an SOC may include a processor core, memory controller, video components, audio components, and/or communication components on a single chip.” '838 patent at 1:27-32. Therefore, <i>Tkacik</i>’s secure processing system 12 is a “system-on-a-chip apparatus.” Additionally, the host device 10 (also named as encryption apparatus 14) of <i>Tkacik</i>, which incorporates the SoC secure processing system 12 also constitutes a “system-on-a-chip apparatus” as claimed to the extent that Daedalus argues such view. For example, <i>Tkacik</i> discloses “a host device 10

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	that utilize[s] a secure processing system 12. Secure processing system 12 is an encryption apparatus 14 because it performs data encryption and/or decryption operations. Likewise host device 10 is an encryption apparatus 14 because it performs data encryption and/or decryption operations.” <i>Tkacik</i> ¶ [0020]. Because the host device 10 is a device with the system-on-a-chip 12 as shown in Figure 1 below, the host device 10 reads on the claimed “system-on-a-chip apparatus.”  FIG. 1 <i>Tkacik</i>, Fig. 1. <i>Tugenberg</i> teaches this preamble by disclosing a secure processing system 14, which maps to the claimed “system-on-a-chip apparatus.”

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	 FIG. 1 <i>Tugenberg</i> at Fig. 1 <i>Tugenberg</i> explains that its secure processing system 14 includes “a host processor 16, secure memory 20, bus 18 coupling the host processor and the secure memory and on-chip system components 15,” which “are fabricated on a single integrated circuit chip.” <i>Tugenberg</i> at 2:43-51; <i>see also Tugenberg</i> at 2:33-39, 1:26-36, 1:40-55, 1:59-61, 2:3-21, 2:31-58, 4:20-31, 4:54-58, 5:48-53, 6:12-20, 8:7-20, claims 6, 14, Fig. 1, Abstract. The description that all the components of secure processing system 14 “are fabricated on a single integrated circuit chip” is consistent with the description in the specification of the '838 patent. <i>See</i> '838 patent at 1:27-32. Therefore, <i>Tugenberg</i>’s secure processing system 14 is a “system-on-a-chip apparatus.” Alternatively, to the extent that Daedalus argues a device incorporating a system-on-a-chip is the claimed “system-on-a-chip apparatus”, the entire host device 10 could also be mapped as the claimed “system-on-a-chip apparatus.” For example, <i>Tugenberg</i> states: “Communication device 10 is

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	comprised of a secure processing system 14 having an associated external memory 12 and other communication components 13. Device 10 may, for example, be a wireless communication device, such as a cellular or wireless phone, or a wireline communication device such as a computer, or a portion thereof.” <i>Tugenberg</i> at 2:33-39. Because the host device 10 is a device with the system-on-a-chip 14, as Figure 1 shows above, the host device 10 reads on the claimed “system-on-a-chip apparatus” under such view. <i>Paaske</i> also discloses a system-on-a-chip apparatus. <i>Paaske</i>’s Figure 2 illustrates a trusted platform module (“TPM”) “implemented in an integrated circuit 204.” <i>Paaske</i> ¶ [0032]; <i>see also id.</i> ¶ [0007] (describing “a TPM (e.g., an integrated circuit incorporating TPM hardware and code)”), Abstract (stating “a chip may include a TPM”), ¶ [0046] (explaining “the TPM 302 . . . may be implemented within a single integrated circuit (i.e., chip)”), ¶ [0107] (describing “a chip implementing a TPM”). <i>Paaske</i> also explains that such an integrated circuit “may include other components,” such as “one or more cryptographic processor(s) 220, . . . , one or more other processing component(s) 222, . . . , a flash controller 224” so as to “reduce the number of integrated circuits on the motherboard.”), ¶ [0220] (“Such components may be implemented on one or more integrated circuits. For example, in some embodiments several of these components may be combined within a single integrated circuit. In some embodiments some of the components may be implemented as a single integrated circuit.”); ¶ [0221] (“In some embodiments some of the connections represented by the lead lines in the drawings may be in an integrated circuit, on a circuit board.”), Abstract, ¶ [0004], ¶ [0007], ¶¶ [0023]-[0026], ¶ [0032], ¶ [0034], ¶¶ [0036]-[0037], ¶¶ [0043]-[0044], ¶ [0046], ¶ [0056], ¶ [0060], ¶ [0075], ¶¶ [0082]-[0083], ¶¶ [0085]-[0086], ¶¶ [0098]-[0099], ¶ [0102], ¶ [0111], ¶ [0117], ¶¶ [0126]-[0127], ¶ [0130], ¶ [0159], ¶ [0172], ¶ [0181], ¶ [0190], ¶ [0205], ¶ [0208], ¶ [0212], ¶¶ [0218]-[0227], Figs. 2-3. Such an integrated circuit (e.g., integrated circuit 204) constitutes a “system-on-a-chip apparatus,” as claimed.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	FIG. 2 <i>Paaske</i> at Fig. 2. <i>Paaske</i>'s explanation that a single integrated circuit chip 204 that incorporates a TPM (trusted platform module, e.g., TPM 202 or 302) and other components is consistent with the '838 patent's description. See '838 patent at 1:27-32. Additionally, to the extent that Daedalus argues a device incorporating the system-on-a-chip constitute the claimed "system-on-a-chip apparatus," the "computing system 200" illustrated in <i>Paaske</i>'s Figure 2, which incorporates integrated circuit 204 (including a TPM, such as TPM 202 or 302) also constitutes a "system-on-a-chip apparatus." For example, <i>Paaske</i> explains that "Fig. 2 illustrates one embodiment of a computing system 200 that incorporates a TPM 202. In this embodiment, the TPM 202 is implemented in an integrated circuit 204." <i>Paaske</i> ¶ [0032]; see also <i>id.</i> ¶ [0046] ("In some

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	embodiments [of a secure data processing system 300,] the TPM 302 (and optionally the flash controller 304) may be implemented within a single integrated circuit (i.e., chip)”), Figs. 2-3. Because computing system 200 comprises integrated circuit 204, as Figure 2 shows above, computing system 200 reads on the claimed “system-on-a-chip apparatus.” In addition, the secure data processing system 300, which includes the chip (including TPM 302 and flash controller 304) as Figure 3 shows above, is also the “system-on-a-chip apparatus.” <i>Aaraj</i> teaches this preamble by disclosing “[w]e then present a suite of hardware and software enhancements to accelerate these commands — generic custom instructions and exploitation of parallel processing capabilities in multiprocessor system-on-chip (SoCs).” <i>Aaraj</i> at 8:2-8:3; <i>see also Aaraj</i> at 8:19-8:20 (“generic design flow and SoC template”); 8:30 (“various architectural enhancements that enable a more efficient deployment of ECC-based TPM in embedded SoCs”), 8:1, 8:3, 8:7. In addition, <i>Aaraj</i> describes performing experimental evaluation in a state-of-the-art embedded SoC processor Xtenxa from Tensilica. <i>Aaraj</i> at 8:4; <i>see also Aaraj</i> at 8:2, 8:16, 8:20-8:22, 8:26-28, 8:31. <i>Mujtaba</i> teaches this preamble by illustrating in Figure 9, “a processing system 911, which may be one or more microprocessors, or which may be a system on a chip of integrated circuit.” <i>Mujtaba</i> ¶ [0058]. The processing system 911 may comprise processors such as wireless communication processor 101 and application processor (e.g., general purpose processor) 102 illustrated in Figure 1. <i>See Mujtaba</i> ¶¶ [0026]-[0034]; <i>see also Mujtaba</i> ¶¶ [0063]-[0068], Fig. 9. <i>Gibod</i> teaches this preamble by disclosing “[l]everaging integral secure components of a SoC can simplify and optimize the development process.” <i>Gibod</i> ¶¶ [0018]-[0020] (emphases added), ¶ [0016], ¶¶ [0023]-[0024], ¶ [0037], ¶ [0039], ¶ [0046], ¶ [0065], ¶¶ [0069]-[0070], Figs. 1 and 5; <i>Buer754</i> teaches this preamble by disclosing “a variety of hardware and software processing components may be used to implement the functions These components may be combined on one or more integrated circuits.” <i>Buer754</i> at 16:36-46 (emphases added); <i>see also Buer754</i> at 16:47-59

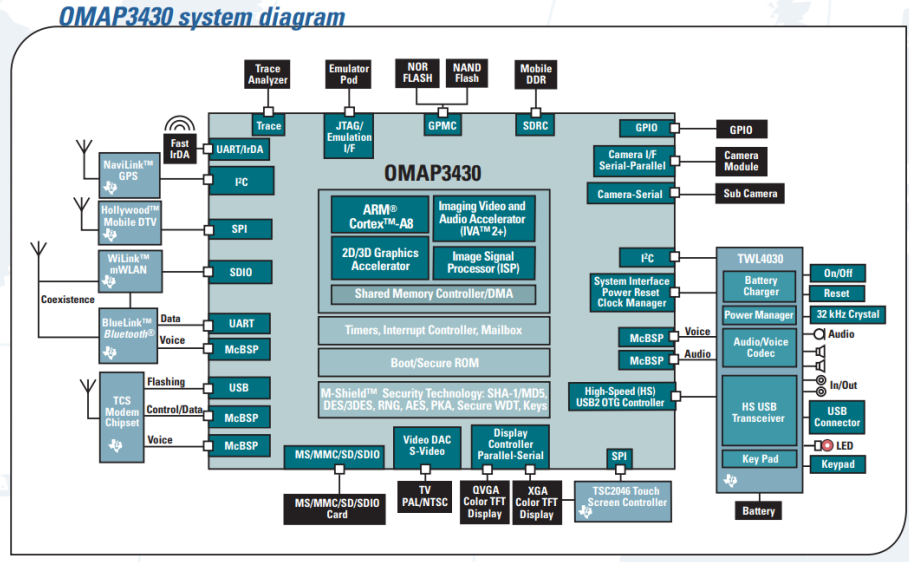
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	(“Some of the connections represented by the lead lines in the drawings maybe in an integrated circuit”); 17:1-62, 4:45-60, 5:52-62, 11:62-12:3, 13:38-44, claims 3-5, Figs. 1-4, 10. Exemplary disclosures from the prior art include the following: <i>Bugiel</i> discloses the following at 220 (citations omitted): <i>Contribution.</i> In this paper, we present TruWalletM, a hardware-assisted wallet-based web authentication as a mechanism to protect user credentials from malware, phishing and physical attacks on mobile platforms. We present the design and implementation proposal of our solution, which also meets important usage requirements: (i) compatibility with existing password-based web authentication, (ii) software reuse of legacy OS and legacy web-browser on smartphones, and (iii) not noticeable performance overhead. To the best of our knowledge, no other solution exists for mobile phones that meets these requirements. Our key design solution, which allows us to meet these requirements, is to split a single SSL/TLS connection between the user device and the server into two logically separated channels, where one is protected by TrEE and is used to transmit passwords, and another one is intended for conventional data. Our design relies on the availability of a Trusted Execution Environment (TrEE) to protect the execution of critical code from tampering. Such TrEEs can be provided by commodity secure hardware for mobile devices such as M-Shield [8] and ARM TrustZone [9]. <i>See also Bugiel</i> at Title, 219-225, 227-236. <i>Chou</i> discloses the following at ¶¶ [0065]-[0066]: HSM 20 may be implemented in a variety of components, circuits, devices, processors, state machines, programmable arrays, etc. It is to be noted that HSM 20 may be constructed on a single integrated circuit (IC) chip or on multiple chips. In other embodiments, components of device 30 is implemented within a single integrated circuit (IC) chip 91 that

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	incorporates a complete system on the IC chip (System-On-Chip or SOC). FIG. 4 shows one such embodiment in which HSM 20 and other components of FIG. 1 or 3 are implemented as SOC 91. It is to be noted that device 30 of FIG. 3 may be readily implemented as SOC 91. The particular implementation of SOC 91 in FIG. 4 is a wireless device 90 that is used to transmit and receive wireless communication. . . . <i>See also Chou</i> Title, ¶ [0006], ¶¶ [0017]-[0018], ¶ [0020], ¶ [0027], ¶ [0036], ¶ [0040], ¶¶ [0042]-[0055], ¶ [0063], ¶¶ [0065]-[0066], ¶¶ [0068]-[0069], Figs. 1, 3-5. <i>See also Tkacik</i> at Abstract, ¶ [0020], ¶¶ [0001]-[0002], ¶¶ [0011]-[0012], ¶¶ [0015]-[0018], ¶¶ [0022]-[0023], ¶¶ [0025]-[0026], ¶¶ [0029]-[0034], ¶ [0037], ¶¶ [0039]-[0040], ¶ [0046], ¶ [0058], ¶ [0064], ¶ [0066], ¶ [0073], ¶ [0079], ¶ [0089]. <i>See also Tugenberg</i> at 1:26-36, 1:40-55, 1:59-61, 2:3-21, 2:31-58, 4:20-31, 4:54-58, 5:48-53, 6:12-20, 8:7-20, claims 6, 14, Fig. 1, Abstract. <i>See also Paaske</i> ¶¶ [0220]-[0221], Abstract, ¶ [0004], ¶ [0007], ¶¶ [0023]-[0026], ¶ [0032], ¶ [0034], ¶¶ [0036]-[0037], ¶¶ [0043]-[0044], ¶ [0046], ¶ [0056], ¶ [0060], ¶ [0075], ¶¶ [0082]-[0083], ¶¶ [0085]-[0086], ¶¶ [0098]-[0099], ¶ [0102], ¶ [0107], ¶ [0111], ¶ [0117], ¶¶ [0126]-[0127], ¶ [0130], ¶ [0159], ¶ [0172], ¶ [0181], ¶ [0190], ¶ [0205], ¶ [0208], ¶ [0212], ¶ [0218]-[0227], Figs. 2-3. <i>See also Aaraj</i> at 8:1-8:4, 8:7, 8:16, 8:19-8:22, 8:26-28, 8:30-8:31. <i>See also Mujtaba</i> ¶¶ [0026]-[0034], ¶ [0058], ¶¶ [0063]-[0068], Fig. 9. <i>See also Gibod</i> ¶ [0016], ¶¶ [0018]-[0020], ¶¶ [0023]-[0024], ¶ [0037], ¶ [0039], ¶ [0046], ¶ [0065], ¶¶ [0069]-[0070], Figs. 1 and 5.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	See also <i>Buer754</i> at 3:42-48, 4:45-60, 5:52-62, 11:62-12:3, 13:38-44, 16:36-59; 17:1-62, claims 3-5, Figs. 1-4, 10.
[1a-1] a system-on-a-chip comprising a security engine that is separate from a processor core of the system-on-a-chip and	The grounds alleged at the beginning of this chart disclose or suggest “a system-on-a-chip comprising a security engine that is separate from a processor core of the system-on-a-chip.” <i>Bugiel</i> describes an exemplary implementation using Nokia N900 phone, which is based on the SoC OMAP3430. See <i>Bugiel</i> at 230, 219, 234. The SoC OMAP3430 includes M-shield, which is separate from a processor core on the SoC, such as ARM Cortex-A8, etc. as shown below.⁷ <i>Bugiel</i> discloses “the execution environment of a mobile platform is divided into two isolated parts: ‘open world’ and the TrEE” (Trusted Execution Environment). <i>Bugiel</i> at 222-224, FIG. 1.  The diagram, titled "OMAP3430 system diagram", illustrates the internal architecture of the OMAP3430 SoC and its connections to various external components. The central SoC block contains the ARM Cortex-A8 processor, Imaging Video and Audio Accelerator (IVA), 2D/3D Graphics Accelerator, Image Signal Processor (ISP), Shared Memory Controller/DMA, Timers, Interrupt Controller, Mailbox, Boot/Secure ROM, and M-Shield Security Technology (SHA-1/MD5, DES/3DES, RNG, AES, PKA, Secure WDT, Keys). External components connected to the SoC include: Trace Analyzer, Emulator Pod, NOR FLASH, NAND Flash, Mobile DDR, JTAG/Emulation I/F, GPMC, SDRC, GPIO, Camera I/F Serial-Parallel, Camera Module, Sub Camera, TWL4030 (Battery Charger, Power Manager, Audio/Voice Codec, HS USB Transceiver, Key Pad, Battery), TSC2046 Touch Screen Controller, XGA Color TFT Display, OVGA Color TFT Display, TV PAL/NTSC, MS/MMC/SD/SDIO Card, Video DAC S-Video, Display Controller Parallel-Serial, High-Speed (HS) USB2 OTG Controller, McBSP, Audio, Voice, System Interface Power Reset Clock Manager, I2C, Camera-Serial, Camera I/F, SPI, SDIO, UART, IrDA, Fast IrDA, Navilink™ GPS, Hollywood™ Mobile DTV, WiLink™ mWLAN, BlueLink™ Bluetooth, TCS Modem Chipset, Flashing, Control/Data, USB, McBSP, Voice, Data, Voice, Coexistence, and various other interfaces like UART, IrDA, and SPI.

⁷ https://www.ti.com/pdfs/wtbu/ti_omap3430.pdf.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>Bugiel</i> also explains that “the TrEE is isolated from the rest of the system in terms of processing and memory.” <i>Id.</i> at 232. <i>See also Bugiel</i> at 222-224 (disclosing the TruWalletM architecture, which includes a “WalletHelper (in open world) and WalletCore (in TrEE)” and other components of the TrEE (including Secure Storage)), 232 (“the TrEE is isolated from the rest of the system in terms of processing and memory.”), 230, 234-235, FIG. 1. <i>Bugiel</i> also describes the WalletCore can support cryptographic functions, such as those in the SSL/TLS protocol (encryption, hash, etc.). <i>See generally Bugiel</i> on SSL/TLS protocols, e.g., <i>Bugiel</i> at 220-236. The combinations alleged here use one of <i>Chou</i>, <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i>’s hardware to implement <i>Bugiel</i>’s communications method, including to implement secure communication using known protocols, e.g., IKE protocol or SSL/TLS Handshake Protocol, to establish a secure communication with a remote server. Each of <i>Chou</i>, <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i> teaches the SoC comprises a security engine, that is separate from a process core on the SoC. For example, relevant to <i>Bugiel</i>’s disclosure of secure execution environment within SoC, <i>Chou</i>’s Figure 1 illustrates HSM 20 and Key Algorithm Module 15 are separate from CPU 11 (the claimed “processor core”). <i>Chou</i> further explains that SPU-M module 39 of Fig. 3 “is equivalent to key algorithm module of 15 of Fig. 1.” <i>Chou</i> ¶ [0054]. In addition, “[p]rocessor 32 [of FIG. 3] is equivalent to CPU 11 of FIG. 1.” <i>Id.</i> ¶ [0031]. Therefore, Figure 3 also illustrates HSM 20 and SPU-M 39 (together, the claimed “security engine”) are separate from the CPU 32 (the claimed “processor core”). For convenience of discussion, this chart refers to HSM 20 and SPU-M 39 of Fig. 3, collectively, as the claimed “security engine,” with the understanding that HSM 20 and Key Algorithm module 15 of Fig. 1, collectively, also map to the “security engine” because SPU-M module 39 is equivalent to Key Algorithm module 15.

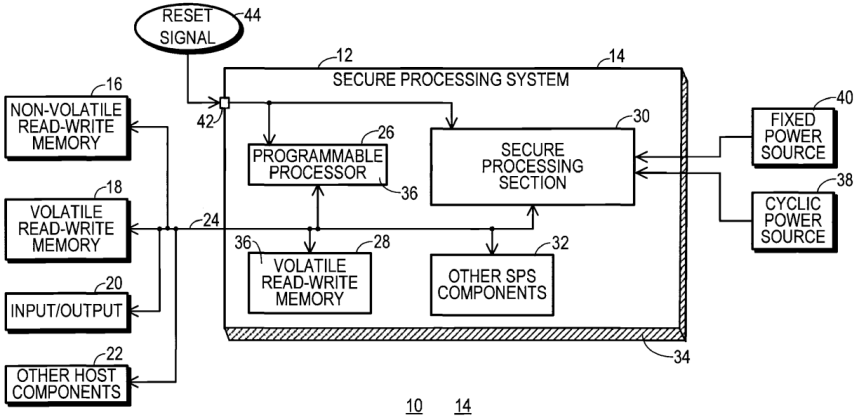
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	FIG. 1 FIG. 3 <i>Chou</i> at Figs. 1, 3 (annotated). As <i>Chou</i>'s Figures 4 and 5 illustrate, "HSM 20 and other components of FIG. 1 or 3 are implemented as SOC 91." <i>Chou</i> ¶ [0065]. Therefore, SOC 91 (the claimed "system-on-a-chip") comprises HSM 20 and SPU-M 39 (collectively, the claimed "security engine") and CPU 11/32 (the claimed "processor core"); <i>see also Chou</i> ¶¶ [0017]-[0018], ¶¶ [0020]-[0025], ¶ [0027], ¶¶ [0029]-[0031], ¶¶ [0034]-[0037], ¶¶ [0054]-[0055], ¶ [0063], ¶ [0065], ¶¶ [0068]-[0071], Fig. 1 and 3-5.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	<i>Chou</i> at Figs. 4-5 (annotated). <i>Chou</i> explains that HSM 20 includes a PKA (Public Key Accelerator) module 56, which is an accelerator operable for the cryptographic key exchange algorithms in IPsec IKE or SSL/TLS protocols, such as Diffie-Hellman public value and shared key computation and RSA key generation. <i>See Chou</i> ¶¶ [0043]-[0053]. <i>Chou</i>'s PKA module is also operable for signing and hashing algorithms like "1024-bit DSA signature with SHA-1 hash or 2048-bit DSA signature with SHA-256 hash." <i>See Chou</i> ¶ [0045]. <i>Chou</i> explains that SoC 91 includes a Key Algorithm module/SPU-M (Secure Processing Unit for Mobile processors) that operates as a hardware accelerator for symmetric key algorithms, "such as AES, DES, RC4, SHA1-HMAC, SHA2-HMAC, MD5-HMAC." etc., for encrypting/decrypting messages using key from key cache 65. <i>Chou</i> ¶¶ [0054]-[0062]. <i>Chou</i> explains that HSM 20 includes a KEK (key-encryption key) unit 55, which wraps and unwraps keys using a root/device unique key programmed in OTP during manufacturing. <i>See Chou</i> ¶ [0023], ¶ [0040], ¶ [0064]. In the alleged combination, the elements of <i>Bugiel</i>'s security engine in the secure execution environment would run on e.g., <i>Chou</i>'s HSM 20 and SPU-M 39, and the non-secure processing and memory for <i>Bugiel</i>'s method would take place in <i>Chou</i>'s components other than HSM 20 and SPU-M 39 (e.g., CPU 32, DMA 33, and/or RAM 35-38).

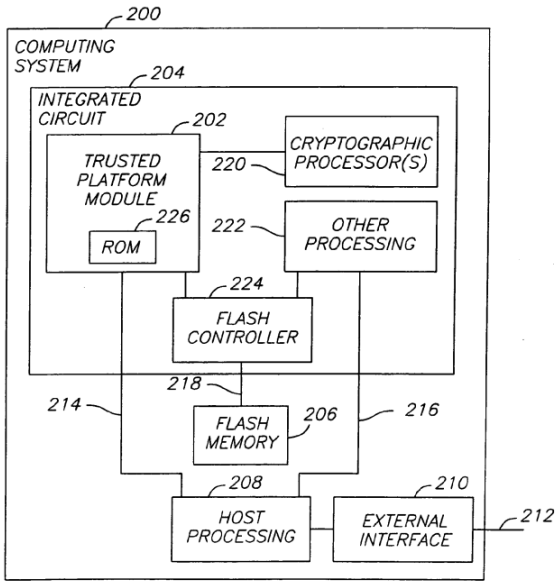
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	In addition, Fig. 1 of <i>Tkacik</i> illustrates the secure processing section 30 is separate from programmable processor 26 of the SoC secure processing system 12; <i>see also Tkacik</i> ¶¶ [0005]-[0008], ¶¶ [0012]-[0013], ¶¶ [0022]-[0026], ¶ [0030], ¶¶ [0033]-[0034], ¶ [0037], ¶ [0039], ¶¶ [0043]-[0044], ¶¶ [0047]-[0049], ¶ [0058], ¶ [0064], ¶¶ [0066]-[0067], ¶¶ [0073]-[0074], ¶ [0079], ¶ [0087], ¶ [0089], Abstract, Figures 1-2.  <i>Tkacik</i>, Fig. 1. <i>Tkacik</i> also describes its hardware is capable of performing the various wireless and cryptographic functions (generating random nonce, signing, hashing, encryption/description, RSA, Diffie-Hellman, IKE, SSL, etc.) relevant to <i>Bugiel</i>, IPsec IKE or SSL/TLS protocols. <i>See Tkacik</i> Abstract, ¶¶ [0002]-[0009], ¶¶ [0019]-[0021], ¶ [0023], ¶ [0026], ¶¶ [0028]-[0029], ¶ [0031], ¶¶ [0037]-[0038], ¶ [0045], ¶¶ [0047]-[0056], ¶¶ [0059]-[0062], ¶¶ [0064]-[0089], Abstract, Figs. 1-3, 5-8. In addition, <i>Paaske</i> teaches this claim feature by illustrating in Figure 2, integrated circuit 204 (a “system-on-a-chip”) comprising a TPM 202 and cryptographic processor(s) 220 (together, the claimed “security engine”) that are separate from other processing 222 (the claimed “processor core of the system-on-a-chip”). <i>See Paaske</i> ¶ [0032] (“[T]he TPM 202 is implemented in an integrated circuit 204.”), ¶ [0034] (“In some embodiments the integrated circuit 204 may include other components to,

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	for example, reduce the number of integrated circuits on the motherboard. For example, one or more cryptographic processor(s) 220 that may be used by the TPM 202 may be incorporated into the integrated circuit. In addition, the integrated circuit 204 may include one or more other processing component(s) 222 that may operate independently of the TPM 202.”); <i>see also Paaske</i> at Abstract, Figs. 2-3. <i>Paaske</i> also teaches that cryptographic processors 220 may be included within the TPM. <i>See id.</i> ¶ [0031] (“[T]he TPM may include one or more cryptographic processors.”). For example, <i>Paaske</i>’s Figure 3 shows cryptographic processors 330 and 334 as being included within TPM 302. <i>See also id.</i> ¶ [0054] (“[T]he instruction cache controller 326 may include one or more cryptographic processor(s) 334 for performing the decryption and authentication operations.”), ¶ [0055] (“The cryptographic processor(s) 330 may be used to provide cryptographic processing for the TPM 302. For example, the cryptographic processor(s) 330 may include a public key core 350 . . . an authentication core 352 . . . [and] a symmetric cryptographic core 354 . . .”). For purposes of this chart, the TPM may include such cryptographic processors and that the TPM itself (e.g., TPM 302) or TPM and cryptographic processors collectively may constitute the claimed “security engine”; <i>see also Paaske</i> ¶ [0007], ¶ [0050], ¶ [0111], ¶ [0227].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	FIG. 2  <i>Paaske</i> at Fig. 2. In addition, <i>Paaske</i> also describes its security engine may perform cryptographic functions relevant in <i>Bugiel</i> or IKE or SSL/TLS protocols. See <i>Paaske</i> at ¶ [0005] (encryption/decryption), ¶¶ [0028]-[0029] (encryption/decryption, authentication), ¶ [0031] (encryption, decryption, authentication, verification, signing and key management), ¶¶ [0042]-[0044] (encryption, authentication), ¶¶ [0055]-[0056] (DSA, RSA, Diffie-Hellman, hashing, symmetric encryption), ¶ [0101] (encryption, signing), ¶ [0106] (RSA), ¶ [0114] (encryption and authentication), ¶ [0154] (random number generator), ¶ [0169] (RSA, digital signing), ¶ [0177] (RSA, digital signing), ¶ [0191] (random number generator), ¶ [0194] (RSA), ¶ [0175] (hash), ¶ [0105] (hash), ¶ [0131] (hash), ¶ [0156] (hash), ¶¶ [0205]-[0206] (RSA), ¶¶ [0208] -[0210] (RSA); see also ¶¶ [0021]-[0024], ¶¶ [0217]-[0227], Figs. 2 and 3.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	In addition, <i>Tugenberg</i>'s Figure 1 illustrates host processor 16 and secure memory 20, which are separate from system processor 15 of the SoC secure processing system 14. <i>See Tugenberg</i> at 5:65-67 (“[T]he secure memory and the host processor are on the same chip.”). <i>Tugenberg</i> explains: “On-chip system components 15 comprise other system processors, memory and other functional elements that are part of communication device 10 to allow for wireless or wireline communication of video, data and/or voice.” <i>Tugenberg</i> at 2:54-58 (emphasis added); <i>see also Tugenberg</i> at 2:31-58, 5:65-67, 8:7-20, Fig. 1. FIG. 1

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>Tugenberg</i> also discloses its host processor may perform cryptographic functions relevant in <i>Bugiel</i>, IKE or SSL/TLS protocols. <i>See Tugenberg</i> at 5:47-54 (SSL), 2:3-21 (encryption logic circuitry), 2:59-3:18 (encryption logic circuitry), 4:20-31 (encryption logic circuitry), 6:50-58 (encryption logic circuitry, secure memory), Fig. 1, claims 9 and 17; <i>see also Tugenberg</i> at Abstract, 1:40-55, 2:3-21, 2:31-4:19, 4:32-36, 4:50-54, 4:59-5:9, 5:47-54, 5:62-6:2, 6:8-25, 6:40-8:20, claims 1-2, 5-6, , 9, 12, 14, 17. In addition, <i>Aaraj</i> teaches this claim feature by disclosing implementing TPM in a “protected execution domain,” in multiprocessor SoC. <i>See Aaraj</i> at 8:1-8:3, 8:7, 8:19-8:21, 8:23, 8:26-8:28, 8:30. Therefore, <i>Aaraj</i> teaches the TPM is separate from a processor core in the multiprocessor SoC. <i>Aaraj</i> also discloses the TPM may perform cryptographic functions relevant in <i>Bugiel</i> or IKE or SSL/TLS protocols. <i>See Aaraj</i> at 8:1 (RSA, cryptography), 8:3-8:5 (RSA, SHA-1, random number generator, HMAC), 8:6-8:9 (random number generator, RSA, symmetric encryption/decryption, HMAC, SHA-1, SSL, cryptographic functions), 8:7 (encryption/decryption, RSA, HMAC, SHA-1), 8:8 (RSA), 8:9 (SSL, RSA, HMAC, SHA-1, cryptographic functions), 8:15 (SSL, RSA, SHA), 8:31 (SSL). In addition, <i>Gibod</i> teaches this claim feature by disclosing a secure processing unit (“SPU”) 518 is separate from processing unit 502 on a SoC in Fig. 5. <i>See Gibod</i> ¶¶ [0065]-[0067]; <i>see also Gibod</i> at Abstract (describing a secure processing domain separate from an open processing domain), ¶¶ [0005]-[0009], ¶ [0016], ¶¶ [0018]-[0020], ¶¶ [0037]-[0039], ¶¶ [0041], ¶ [0047], ¶¶ [0069]-[0070], Figs. 1 (Secure Processor separate from Media Processing, CPU, GPU and other Operating System) and 5, claims 1, 3-5, 14, 16, 18, and 19. <i>Gibod</i> also discloses its security engine may perform cryptographic functions as described in <i>Bugiel</i>, IKE or SSL/TLS protocols. <i>See Gibod</i> at ¶¶ ¶¶ [0008]-[0009] (encryption/decryption), ¶ [0019] (encryption/decryption), [0033]-[0037] (RSA, symmetric encryption/decryption, signing, hashing), ¶ [0052] (random number generator), ¶ [0055] (encryption/decryption), ¶ [0067] (key management), ¶ [0071] (TLS, hash, symmetric encryption, public key exchange), ¶¶ [0070]-[0071], Fig. 1 (RSA, AES, SHA), claim 19 (symmetric and asymmetric cryptographic functions).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	FIG. 5 FIG. 1 <i>Gibod</i>, Figs. 1 and 5. <i>Mujtaba</i> teaches this claim feature by illustrating in Figure 9, “a processing system 911, which may be one or more microprocessors, or which may be a system on a chip of integrated circuit.” <i>Mujtaba</i> ¶ [0058]. The processing system 911 may comprise processors such as wireless communication processor 101, which is separate from a processor core application processor (e.g., general purpose processor) 102 illustrated in Figure 1. <i>See also Mujtaba</i> ¶¶ [0026]-[0034], ¶¶ [0063]-[0068]. <i>Mujtaba</i> also discloses its wireless communication processor may perform cryptographic functions relevant in <i>Bugiel</i>, IKE or SSL/TLS protocols. <i>See Mujtaba</i> at ¶ [0006] (session key generation, encryption/decryption, key management), ¶¶ [0042]-[0044] (cryptographic unit for encryption), ¶¶ [0052]-[0053] (session key generation, encryption/decryption, key management), ¶ [0054] (hash),

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	¶ [0039] (encryption), ¶¶ [0047]-[0050] (cryptographic unit, encryption), ¶ [0055]-[0057] (encryption/decryption), ¶¶ [0066]-[0068], Fig. 4. In addition, <i>Buer754</i> teaches this claim limitation. For example, Fig. 3 of <i>Buer754</i> discloses security module 320 and cryptographic accelerator 326 (collectively the security engine) are separate from host processor 322. <i>Buer754</i> at 6:13-23; <i>see also Buer754</i> at 4:61-5:6, 6:13-35, 10:14-28, 11:62-12:3, 17:48-62. <i>Buer754</i> also discloses the security engine may perform cryptographic functions relevant in <i>Bugiel</i>, IKE or SSL/TLS protocols. <i>See Buer754</i> at 1:26-2:13 (SSL, IPsec IKE, RSA key exchange, encryption/decryption), 6:13-31 (SSL, public key exchange), 8:63-9:4 (IKE), 13:14-15:11 (Diffie-Hellman, RSA, HMAC), 15:17-16:35 (IPsec, HMAC, SHA1, SSL/TLS, RSA, DSA, Diffie-Hellman, 3DES, AES); <i>see also Buer754</i> at 5:24-29, 9:16-21. In addition, at the time of the '838 patent, it was common knowledge to a POSITA that a security engine should be separate from a processor core to ensure the security of the cryptographic processing of sensitive information. For example, <i>Weise</i>⁸ describes a composite hardware security module (HSM) cryptographic accelerator 100 should be “without the intervention of the host operating system.” <i>Weise</i> ¶ [0027], Fig. 1; <i>see also Weise</i> Abstract, ¶ [0002], ¶¶ [0005]-[0006], ¶¶ [0008]-[0013], ¶¶ [0016]-[0020], ¶ [0023]-[0043], Fig. 1, claims 1-12, 14-17, and 19. <i>Weise</i> also discloses the HSM may perform cryptographic functions such as exponential key exchange, AES, DES, 3DES, RSA, DSA, MD5, hash, RNG, ECC, SSL, IPsec etc. <i>See Weise</i> ¶ [0002], ¶¶ [0005]-[0006], ¶¶ [0008]-[0013], ¶¶ [0016]-[0020], ¶¶ [0023]-[0041], Abstract, Figs. 1, 3-5, claims 1-12, 14-17, 19; <i>see also Davis</i>⁹ at Figs. 1-4, 6 (the chipset 315 and cryptographic unit 335 (together forming the “security engine”) are separate from host processor), 9 (chipset with bulk cryptographic functionality and generation cryptographic operations 910 is separate from host processor 205), 10, Abstract, 1:28-43, 2:5-25, 2:32-42, 2:48-62, 2:65-3:3, 3:13-15, 3:46-4:36, 5:13-32, 3:45-53, 6:36-65, 8:7-20, claims 1, 13, 17, 25, 29.

⁸ U.S. Patent Application Publication No. 2008/0181399 A1 to Weise et al (“*Weise*”), which was filed on January 29, 2007, and published July 31, 2008, qualifies as prior art under at least pre-AIA 35 U.S.C. §§ 102(a), (b) and (e).

⁹ U.S. Patent No. 5,818,939 to Davis (“*Davis*”).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Exemplary disclosures from the prior art include the following: <i>Bugiel</i> discloses the following at 232 (Security Considerations): Protection of user credentials. The user credentials are protected by the following means: (i) run-time isolation. All operations on user credentials (except user input) are performed within the TrEE. As the TrEE is isolated from the rest of the system in terms of processing and memory, that guarantees protection of the user's credentials at run-time from potentially malicious OS and other OS-side components. . . . <i>See also Bugiel</i> at 219-236. <i>See also Chou</i> ¶¶ [0017]-[0018], ¶¶ [0020]-[0025], ¶ [0027], ¶¶ [0029]-[0031], ¶¶ [0034]-[0037], ¶ [0040], ¶¶ [0043]-[0065], ¶¶ [0068]-[0071], Fig. 1 and 3-5. <i>See also Tkacik</i> ¶¶ [0002]-[0009], ¶¶ [0012]-[0013], ¶¶ [0019]-[0026], ¶¶ [0028]-[0031], ¶¶ [0033]-[0034], ¶¶ [0037]-[0039], ¶¶ [0043]-[0045], ¶¶ [0047]-[0056], ¶¶ [0058]-[0062], ¶¶ [0064]-[0089], Abstract, Figures 1-3, 5-8. <i>See also Paaske</i> ¶ [0005], ¶ [0007], ¶¶ [0021]-[0024], ¶¶ [0028]-[0029], ¶¶ [0031]-[0032], ¶ [0034], ¶¶ [0042]-[0044], ¶ [0050], ¶¶ [0054]-[0056], ¶ [0101], ¶ [0105]-[0106], ¶ [0111], ¶ [0114], ¶ [0131], ¶ [0154], ¶ [0156], ¶ [0169], ¶ [0175], ¶ [0177], ¶ [0191], ¶ [0194], ¶¶ [0205]-[0206], ¶¶ [0208] – [0210], ¶¶ [0217]-[0227], Abstract, Figs. 1-3. <i>See also Tugenberg</i> at Abstract, 1:40-55, 2:3-21, 2:31-4:36, 4:50-54, 4:59-5:9, 5:47-54, 5:62-6:2, 6:8-25, 6:40-8:20, Fig. 1, claims 1-2, 5-6, 9, 12, 14 and 17. <i>See also Aaraj</i> at 8:1-9, 8:15, 8:19-8:21, 8:23, 8:26-8:28, 8:30-8:31. <i>See also Gibod</i> at Abstract¶, ¶ [0005]-[0009], ¶ [0016], ¶¶ [0018]-[0020], ¶¶ [0033]-[0039], ¶¶ [0041], ¶ [0047], ¶ [0055], ¶¶ [0065]-[0067], ¶¶ [0069]-[0071], Figs. 1 and 5, claims 1, 3-5, 14, 16, 18, and 19.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also Mujtaba</i> ¶ [0006], ¶¶ [0026]-[0034], ¶ [0039], ¶¶ [0042]-[0044], ¶¶ [0047]-[0050], ¶¶ [0052]-[0058], ¶¶ [0063]-[0068], Figures 1, 4, and 9, claim 8. <i>See also Buer754</i> at 1:26-2:13, 3:42-48, 4:61-5:6, 5:24-29, 6:13-35, 8:63-9:4, 9:16-21, 10:14-28, 11:62-12:3, 13:14-15:11, 15:17-16:35, 17:48-62, Figs. 2-3, 10. <i>See also supra</i> [1pre].
[1a-2] has a secure memory accessible only by the security engine,	The grounds alleged at the beginning of this chart disclose or suggest the security engine “has a secure memory accessible only by the security engine.” <i>Chou</i> teaches the claimed “secure memory”—specifically, the memories located within secure zone 80 illustrated in Figs. 1 and 3 and memories associated with processing within security engine, e.g., security processor, PKA, KEK, etc. As <i>Chou</i>’s Figure 3 indicates, memories, e.g., key cache 65, OTP (One Time Programmed memory) 54, HSM BROM 52, and HSM RAM 53, etc., all reside within “dashed-line box 80, [which] represents the actual secure domain boundary that is not accessible by non-secure components.” <i>Chou</i> ¶ [0034]; <i>see also Chou</i> at Fig. 1.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	FIG. 3 In addition, <i>Chou</i>'s Figure 1 indicates that memories, e.g., key cache 25, secure key module 23, and secure memory 22, etc., are located within HSM module 20. <i>See id.</i> ¶ [0021], ¶¶ [0024]-[0026]. As <i>Chou</i> explains, key cache 25 “resides within the secure domain.” <i>Id.</i> ¶ [0026]. Secure key module 23 is “a storage component in the non-secure domain.” <i>Id.</i> ¶ [0024]. And, as its name indicates, secure memory 22 is also a secure memory. <i>See id.</i> ¶ [0021], ¶ [0023].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	FIG. 1 All of these memories (e.g., secure memory 22, secure key 23, key cache 25/65, OTP 54, HSM BROM 52, and HSM RAM 53, etc.) reside within the HSM 20 and SPU-M 39 and are only accessible only by HSM 20 and SPU-M 39. <i>See Chou</i> at Figs. 1 and 3. Collectively, these memories are the claimed “secure memory” accessible only by the claimed “security engine” (e.g., HSM 20 and SPU-M 39, collectively). <i>See also Chou</i> ¶ [0003], ¶ [0012], ¶ [0018], ¶¶ [0037]-[0041], ¶¶ [0070]-[0071], claims 1-3, 6-7, 12, 14, 17-18, Abstract. These disclosures are also consistent with <i>Bugiel</i>’s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. <i>See Bugiel</i> at 222-225, 228, 231-234. One of the assumptions underlying <i>Bugiel</i>’s TruWalletM method is “availability of a secure hardware which provides a Trusted Execution Environment (TrEE) with the following feature . . . (ii) secure

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	storage.” <i>Bugiel</i> at 222. <i>Bugiel</i> further explains: “SecureStorage is a database where passwords and other security sensitive data are stored persistently and securely.” <i>Bugiel</i> at 224. As <i>Bugiel</i>’s Figure 1 shows, such secure storage is only accessible by the WalletCore in the TrEE. And, according to <i>Bugiel</i>, the SecureStorage is isolated “from other, unauthorized (and potentially malicious) scripts.” <i>Id.</i> at 231; <i>see also Bugiel</i> at 222-225, 228, 231-235. Therefore, a POSITA would have found it obvious to use <i>Chou</i>’s secure memories (e.g., one or more of <i>Chou</i>’s Secure Memory 22, Secure Key 23, Key Cache 25/65, OTP 54, HSM BROM 52, and HSM RAM 53, etc.) to implement <i>Bugiel</i>’s secure storage, e.g., the claimed secure memory. In addition, <i>Tkacik</i> teaches the claimed “secure memory” as memory located within secure processing section 30. For example, <i>Tkacik</i> describes “memory located within secure processing section 30 (discussed below) is secured memory.” <i>Tkacik</i> ¶ [0026]. More specifically, Figure 2 of <i>Tkacik</i> illustrates the following memories are located within the secure processing section 30 (the claimed “security engine”): zeroizable read-write memory 52, key register 54, long-lived non-volatile key register 60, active key register 72, control register 74, data register 76, etc. (together forming the claimed “secure memory”). <i>See id.</i> ¶¶ [0036]-[0038], ¶¶ [0044]-[0046], ¶¶ [0054]-[0055], Fig. 2. <i>Tkacik</i> also explains that key register 54, active key register 72, control register 74, and data register 76 can be included as part of read-write memory 52. <i>See id.</i> ¶ [0038], ¶ [0054]. These memories all reside within the secure processing section 30. Therefore, these memories are the claimed “secure memory accessible only by” the secure processing section 30 (the claimed “security engine”); <i>see also id.</i> ¶¶ [0005]-[0008], ¶¶ [0036]- [0039], ¶¶ [0041]-[0042], ¶¶ [0044]-[0049], ¶¶ [0052]-[0055], ¶ [0059], ¶¶ [0067]-[0068], ¶¶ [0070]-[0071], ¶¶ [0074]-[0075], ¶ [0077], ¶¶ [0080]-[0084], ¶ [0087], ¶ [0089], claims 1-2, 8-9, 10-11, 13-14, 16, 18, Figure 2. <i>Tkacik</i> further explains contents in these memories “are protected against the execution of malicious software within host device 10.” <i>Tkacik</i> ¶ [0037], ¶ [0029]. In addition, the security processing section is design to detect tamper attempt for keeping the operations and data within the secure processing section safe. <i>See Tkacik</i> ¶¶ [0039]-[0043], ¶ [0045], ¶ [0061], ¶ [0069], ¶ [0072]. This is consistent with <i>Bugiel</i>’s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. <i>See Bugiel</i> at 222-225, 228, 231-234.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	In addition, <i>Paaske</i> teaches this claim feature. <i>Paaske</i> teaches the claimed “secure memory”—specifically, secure memories located within the TPM (e.g., data buffer 318 (including persistent memory, volatile memory, stack/heap space, and I/O space) and one-time programmable memory 332, as shown in Fig. 3, as well as any memories associated with the TPM’s cryptographic processor 330, cryptographic processors 334, or processor 308). See <i>Paaske</i> ¶¶ [0023]-[0028], ¶ [0043], ¶¶ [0045]-[0048], ¶ [0056], ¶ [0078], ¶ [0080], ¶ [0082], ¶¶ [0110]-[0111], ¶ [0122], ¶¶ [0217]-[0227], Figs. 1-3.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	As <i>Paaske</i>'s Figure 3 indicates, all of these memories reside within the TPM (302) and thus can be accessible only by the TPM. Collectively, these memories are the claimed "secure memory" accessible only by the claimed "security engine." <i>Paaske</i> also describes the TPM provides "secure protection" of the secure execution environment. <i>See Paaske</i> ¶ [0007]-[0008], ¶ [0056]. This is consistent with <i>Bugiel</i>'s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. <i>See Bugiel</i> at 222-225, 228, 231-234. In addition, <i>Tugenberg</i>'s secure memory 20 constitutes a "secure memory" as claimed. <i>Tugenberg</i> explains, for example, that "a secure memory embedded in the Host processor is used to secure the user's sensitive information." <i>Tugenberg</i> at 4:59-61; <i>see also id.</i> at Abstract, 1:7-10, 1:48-54, 2:3-21, 2:31-3:18, 3:34-37, 3:44-4:9, 4:59-61, 5:62-6:2, 6:12-20, 6:50-58, 8:1-20, Fig. 1. A POSITA would have understood these memories are "secure memory accessible only by" host processor 16 and secure memory 20 (together, the claimed "security engine"). <i>Tugenberg</i> explains, for example, that "[s]ecure memory 20 includes system monitor 25" that "functions to help ensure the proper operation of secure memory 20. For example, improper attempts to access laser-scribed encryption key 21 or memory 24 are detected by the system monitor 25." <i>Id.</i> 3:58-62. <i>Tugenberg</i> also describes its apparatus including the secure memory "is equally suitable for the safeguarding and use of any sensitive information." <i>Id.</i> at 4:50-53, 1:40-47 ("a method and system that securely stores user authentication information and other sensitive information so that it cannot be easily extracted from the portable communication device, copied to another device, or modified."), 5:62-6:2 ("the sensitive information is never accessible outside of the host processor and the Secure Memory after it is originally loaded . . . the sensitive information is never available outside of this chip, thus protecting the information from any external attempt to intercept it."); <i>see also id.</i> 1:15-25, 1:48-55, 2:3-30, 4:59-5:9. This is consistent with <i>Bugiel</i>'s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. <i>See Bugiel</i> at 222-225, 228, 231-234. In addition, <i>Aaraj</i> teaches this claim feature by disclosing the security engine TPM provides "capabilities for secure storage." <i>See Aaraj</i> at 8:2; <i>see also Aaraj</i> at 8:3, 8:5-8:7, 8:15, 8:30. <i>Aaraj</i> further discloses "[p]rotected storage is accessible only within the TPM, and is shielded from physical attacks." <i>Aaraj</i> at 8:5. This is consistent with <i>Bugiel</i>'s assumption of secure storage for keeping the

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. <i>See Bugiel</i> at 222-225, 228, 231-234. In addition, <i>Gibod</i> teaches this claim feature by disclosing “protected memory space of the security processor.” <i>Gibod</i> ¶ [0019]; <i>see also Gibod</i> ¶¶ [0004]-[0005] (“a secure processing domain may be established where secret information is securely processed and protected from exposure to applications executing outside the secure processing domain”), ¶ [0067] (“the SPU 518 may include internal memory”), ¶ [0023] (“hardware security element (e.g., One Time Programmable [‘OTP’] memory”), ¶¶ [0005]-[0007], ¶ [0016], ¶¶ [0025]-[0026], ¶ [0046], ¶ [0039], ¶ [0041], ¶ [0049], ¶ [0053], ¶ [0058], ¶ [0067], ¶¶ [0069]-[0070], claim 1. This is consistent with <i>Bugiel</i>’s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. <i>See Bugiel</i> at 222-225, 228, 231-234. In addition, <i>Mujtaba</i> teaches this claim feature by illustrating secure memory accessible only by the security engine wireless communication processor 101, including e.g., public key storage 119, secure ROM 112, internal RAM 113, storage key 114, RAM 103, FS Driver 153 etc. in Figures 1 and 4. <i>See also Mujtaba</i> ¶ [0028], ¶ [0031], ¶ [0033], ¶ [0037], ¶ [0044], ¶¶ [0064]-[0068], Figure 4. This is consistent with <i>Bugiel</i>’s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. <i>See Bugiel</i> at 222-225, 228, 231-234. In addition, <i>Buer754</i> teaches this claim feature. For example, Figure 3 illustrates secure memory accessible only by security module 320 and cryptographic 326 (collectively the security engine), including EEPROM 336, KEK 332, private keys 330, etc. In addition, Figure 4 illustrates secure memory accessible only by the cryptographic accelerator 420 (part of the security engine), including e.g., memories 432A and 432B, EEPROM 434, etc. Figure 5 illustrates secure memory accessible only by cryptographic accelerator 520 (part of the security engine), including e.g., RD_EEPROM, WR_EEPROM, data storages 532, 534 and 536 etc. <i>See also Buer754</i> at 2:17-60, 4:45-60, 5:52-62, 7:59-64, 8:51-56, 10:29-41, 12:45-61, 13:66-14:6, 15:17-32, 15:46-60, 16:63-67, 17:1-62, Figs. 1-5 and 10, claims 1, 9-11, 13. <i>Buer754</i> also describes protecting keys in the secure memory. <i>See Buer754</i> at 3:49-57 (“the keys are stored in a ‘protected portion’ of the system S”), 5:52-62 (“unencrypted security associations never leave the protected cryptographic accelerator 236”), 15:4-11 (“The shared

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	secret must be protected within the cryptographic boundary”); <i>see also Buer</i>754 at 10:29-41, 12:24-39, 14:35-42. This is consistent with <i>Bugiel</i>’s assumption of secure storage for keeping the sensitive data (such as random nonce, encrypted session key and pre-master key, etc.) secure. <i>See Bugiel</i> at 222-225, 228, 231-234. In addition, at the time of the '838 patent, it was common knowledge for a POSITA to implement secure memory accessible only by the security engine. For example, <i>Weise</i> teaches this claim feature by disclosing “secure storage of long-term private keys.” <i>Weise</i> ¶ [0036]. <i>Weise</i> further discloses key storage 150 resides inside and only accessible by the composite HSM 100. <i>See Weise</i> ¶ [0037], ¶ [0023], Figure 1. As another example, <i>Weise</i> discloses services library 210 for “possess[ing] data necessary for the implementation of various cryptographic functionality,” which resides inside and only accessible by the composite HSM 100. <i>Weise</i> ¶ [0028]; <i>see also Weise</i> Abstract, ¶¶ [0010]-[0012], ¶¶ [0023]-[0024], ¶ [0028], ¶ [0032], ¶¶ [0036]-[0037], ¶¶ [0039]-[0040], ¶¶ [0042]-[0043], Figures 1-2, claims 1, 7, 12-13, and 19. Exemplary disclosures from the prior art include the following: <i>Bugiel</i> discloses the following at 222: Assumptions. We rely on availability of a secure hardware which provides a Trusted Execution Environment (TrEE) with following features: (i) isolated secure code execution, (ii) secure storage, (iii) integrity protection of secure execution environment. Such TrEEs can be provided by general purpose secure hardware such as M-Shield [8] and ARM TrustZone [9]. We assume the TrEE provided by secure hardware is tamper-protected. <i>See also Bugiel</i> at 222-225, 228, 231-236, Figure 1. <i>See also Chou</i> ¶ [0003], ¶ [0012], ¶¶ [0017]-[0018], ¶ [0021], ¶¶ [0023]-[0027], ¶ [0034], ¶¶ [0037]-[0041], ¶¶ [0054]-[0062], ¶¶ [0070]-[0071], claims 1-3, 6-7, 12, 14, 17-18, Abstract, Figs. 1, 3.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also Tkacik</i> ¶¶ [0005]-[0008], ¶ [0026], ¶¶ [0036]-[0049], ¶¶ [0052]-[0055], ¶ [0059], ¶ [0061], ¶¶ [0067]-[0069], ¶¶ [0070]-[0072], ¶¶ [0074]-[0075], ¶ [0077], ¶¶ [0080]-[0084], ¶ [0087], ¶ [0089], claims 1-2, 8-9, 10-11, 13-14, 16, 18, Figure 2. <i>See Paaske</i> ¶ [0007]-[0008], ¶¶ [0023]-[0028], ¶ [0043], ¶¶ [0045]-[0048], ¶ [0051], ¶¶ [0055]-[0056], ¶ [0078], ¶ [0080], ¶ [0082], ¶¶ [0110]-[0111], ¶ [0122], ¶¶ [0217]-[0227], Figs. 1-3. <i>See also Tugenberg</i> 1:7-10, 1:15-25, 1:40-55, 2:3-3:18, 3:34-37, 3:44-4:9, 4:50-53, 4:59-5:9, 5:62-6:2, 6:12-20, 6:50-58, 8:1-20, Abstract, Fig. 1. <i>See also Aaraj</i> at 8:2-8:3, 8:5-8:7, 8:15, 8:30-8:31. <i>See also Gibod</i> ¶¶ [0004]-[0007], ¶ [0016], ¶¶ [0019], ¶ [0023], ¶ [0025]-[0026], ¶ [0039], ¶ [0041], ¶ [0046], ¶ [0049], ¶ [0053], ¶ [0058], ¶ [0067], ¶¶ [0069]-[0070], claim 1. <i>See also Mujtaba</i> ¶ [0028], ¶ [0031], ¶ [0033], ¶ [0037], ¶ [0044], ¶¶ [0064]-[0068], Figs. 1 and 4. <i>See also Buer754</i> at 2:17-60, 3:42-3:57, 4:45-60, 5:52-62, 7:59-64, 8:51-56, 10:29-41, 12:24-39, 12:45-61, 13:66-14:6, 14:35-42, 15:4-11, 15:17-32, 15:46-60, 16:63-67, 17:1-62, Figs. 1-5 and 10, claims 1, 9-11, 13. <i>See supra</i> [1a-1].
[1a-3] wherein the secure memory includes a security key that was encoded in the secure memory during a manufacturing process of the system-on-a-chip,	The grounds alleged at the beginning of this chart disclose or suggest “the secure memory includes a security key that was encoded in the secure memory during a manufacturing process of the system-on-a-chip.” <i>Bugiel</i> describes a key (the claimed “security key”) that is used to encrypt session key and never available outside the TrEE: “the SSL/TLS session key <i>Sc</i> is generated . . . and sealed (together with pre-master secret PMS and the hash value hash(<i>Certs</i>)).” <i>Bugiel</i>, at 225. <i>Bugiel</i> explains that “[s]ealing means protecting an object so that only a certain set of entities can access or use it,” <i>id.</i> at 225 & n. 8,

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	and that, “[w]hen sealed, data is encrypted under a key that is never available outside the TrEE,” <i>id.</i> at 231; <i>see also Bugiel</i> at 226, 228-230. Therefore, <i>Bugiel</i> discloses the claimed “security key” by describing a key that is never available outside the TrEE and that is used to encrypt the SSL/TLS session key Sc. A POSITA would have found it obvious to use the security key (or under Dadelus’s apparent view, key(s) derived from the security key) disclosed in alleged grounds to encrypt the cryptographic keys generated in the secure communication using known protocols (e.g., session key, pre-master key, etc.) in <i>Bugiel</i>. For example, <i>Chou</i> discloses a device unique key/root key that is never available outside of secure domain 80 and can be used to encrypt security keys such as a session key or pre-master key, etc. <i>Chou</i> explains that its device unique key/root key is used for encrypting other security keys. <i>See Chou</i> ¶ [0040] (“KEK unit 55 provides an ability to wrap/unwrap keys using AES key wrap specification and has an ability to load the root key directly from OTP 54”), ¶ [0037] (“Components 54-59 are operable to provide processing operations that pertain to secure key functions, including the wrapping and unwrapping of keys, context data, etc.”), ¶ [0069] (explaining “[s]ecure keys are wrapped/unwrapped and authenticated within the secure zone and do not leave the HSM in an unwrapped state”). For example, <i>Chou</i> teaches using the device unique key/root key to wrap other keys via the AES key wrap specification. <i>Chou</i> ¶ [0040]; <i>see also id.</i> ¶ [0007], ¶¶ [0023]-[0025]. <i>Chou</i>’s Figure 3 below shows PKA module 56 and KEK Unit 55 are communicatively coupled via bus 71. <i>See Chou</i> ¶ [0035] (“Other units 54-69 are also coupled to bus 71”). Additionally, “a direct link exists between KEK Unit 55 and OTP so that the key stored in OTP 54 may be loaded into KEK unit 55 without transitioning onto bus 71.” <i>Id.</i> ¶ [0039]. Using these connections, <i>Chou</i>’s KEK unit 55 loads the session key generated by PKA module 56 and the device unique key/root key from OTP 54 to encrypt the session key using the device unique key/root key. <i>See id.</i> ¶¶ [0039]-[0040], Fig. 3.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	FIG. 3 Because <i>Chou</i> discloses a device unique key/root key to encrypt other keys based on AES encryption algorithm, a POSITA would have found it obvious to use it to seal the session key (Sc) together with the pre-master secret (PMS) and the hash value (hash(Certs)), as <i>Bugiel</i> describes. See <i>Chou</i> ¶ [0040]; <i>Bugiel</i> at 225.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>Chou</i> further explains that its device unique key/root key “may be programmed and stored” in OTP 54. <i>Chou</i> ¶ [0038], ¶ [0064]. A POSITA would have understood that the device unique key/root key is a security key encoded in OTP 54 (the one-time-programmable memory) at the time of manufacturing because (1) that is what makes it a <i>device unique</i> key/root key; (2) “OTP 54 is a non-volatile memory that is used to one-time program a key value or values for device 30,” <i>id.</i> ¶ [0038]; and (3) <i>Chou</i> describes programming OTP 54 “during manufacturing,” <i>id.</i> ¶ [0064]; <i>see also Chou</i> Abstract, ¶ [0003], ¶ [0005], ¶¶ [0007]-[0009], ¶¶ [0017]-[0019], ¶ [0021], ¶¶ [0023]-[0028], ¶¶ [0034]-[0041], ¶¶ [0054]-[0055], ¶ [0063]-[0064], ¶¶ [0069]-[0071], claims 1, 4, 6, 8, 19, Figs. 1 and 3. In addition, <i>Tkacik</i> teaches this claim feature by disclosing a long-lived key register 60 (part of the claimed “secure memory”) includes a plaintext master key 160 (the claimed “security key”) that was encoded in the long-lived key register 60 during a manufacturing process of the secure processing system 12 (the claimed “system-on-a-chip”). <i>See Tkacik</i> ¶ [0046] (“long-lived key register 60 is desirably programmed with plaintext master key 160 during the manufacturing process of secure processing system 12 and cannot thereafter be altered.”); <i>see also</i> ¶¶ [0002]-[0004], ¶¶ [0006]-[0009], ¶ [0019], ¶ [0029], ¶ [0039], ¶¶ [0044]-[0045], ¶ [0048], ¶ [0057], ¶¶ [0069]-[0070], ¶ [0074], ¶¶ [0083]-[0085], ¶¶ [0088]-[0089], Figure 2, Abstract. <i>Tkacik</i> also describes the secure processing section uses this master key to encrypt other keys generated by the secure processing section. For example, <i>Tkacik</i> also explains that plain master key 160 can “directly serve[] as the BKEK” if desired. <i>Id.</i> ¶ [0083]. Here, BKEK means “blob key encryption key.” <i>Id.</i> ¶ [0069]. The BKEK is a cryptographic key to be used in forming a blob, “a self-contained object which includes a cryptographic key in an encrypted form and a message authentication code (MAC).” <i>Id.</i> ¶ [0070]. <i>Tkacik</i> further teaches generating a plaintext active key 172 using IKE or SSL. <i>Tkacik</i> ¶ [0080]. Referring to process 140 in <i>Tkacik</i>’s Figure 8, <i>Tkacik</i> describes moving the generated plaintext active key 172 “to data register 76 and encrypt[ing] this key 172 using . . . the BKEK” (the plaintext master key 160, i.e., the claimed “security key”). <i>Tkacik</i> ¶ [0084], Fig. 8. According to <i>Tkacik</i>, the secure process section 30 performs this encryption. <i>See Tkacik</i> ¶ [0079] (“Process 140 may be performed by controller 46 in response to the execution of software programming instructions using components of secure processing section 30, and possibly other components of host device 10.”). <i>Tkacik</i>’s description of using the master key to encrypt other keys generated based on IKE, SSL or other similar known protocol for secure communication is consistent with <i>Bugiel</i>’s description of using a key to encrypt the

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	session key and pre-master key. <i>See Bugiel</i> 225-227, 231; <i>see also Tkacik</i> ¶¶ [0002]-[0004], ¶¶ [0006]-[0009], ¶ [0019], ¶ [0029], ¶ [0039], ¶¶ [0044]-[0046], ¶¶ [0049]-[0054], ¶ [0057], ¶¶ [0069]-[0070], ¶ [0074], ¶¶ [0079]-[0080], ¶¶ [0083]-[0085], ¶¶ [0088]-[0089], Figure 2, Abstract. In addition, <i>Paaske</i> teaches this claim feature by disclosing that “when the TPM 302 is manufactured security routines may be used to generate secret keys that are unique to each TPM. These keys may then be stored in the OTP memory 332. Moreover, provisions may be made to ensure that these secret keys are not allowed to be read out of the TPM (or the integrated circuit within which the TPM resides) in the clear (e.g., in unencrypted form).” <i>Paaske</i> ¶ [0056]; <i>see also id.</i> ¶ [0101] (“[A]t some point during the manufacturing process . . . an encryption key . . . may be generated for the code update process.”), ¶ [0108] (“[A]t some point during the manufacturing process (e.g., during a test or boot procedure) the TPM may generate . . . internal keys that may be unique to that specific TPM chip.”). Therefore, <i>Paaske</i> discloses the claimed “security key” by describing “secret keys” that are generated when the TPM is manufactured and stored in the OTP 332 (part of the claimed “secure memory”). <i>See Paaske</i> at ¶¶ [0043]-[0044], ¶¶ [0110]-[0111], ¶¶ [0059]-[0060], ¶¶ [0217]-[0227], Fig. 3. <i>Paaske</i> explains that its secret keys are used for encrypting other keys. <i>See Paaske</i> ¶ [0056] (“Through the use of these keys and/or keys derived from these keys, the TPM may securely store information outside of the TPM (or the integrated circuit within which the TPM resides). For example, the cryptographic processor(s) 330 may use these keys to encrypt and/or authenticate any data that is stored in an external memory such as the flash memory 306.”), ¶ [0110] (“[T]he keys stored in the OTP memory may be used to encrypt other keys so that these other keys may be securely stored in external memory.”). <i>Paaske</i> also explains its secret keys are protected in the OTP of the TPM. Therefore, <i>Paaske</i> teaches a security key (e.g., a unique internal security key) that was encoded in the secure memory (e.g., in OTP 332) during a manufacturing process of the system-on-a-chip (e.g., <i>Paaske</i>’s integrated circuit). <i>Paaske</i>’s description of using key that is not allowed to be read out of the TPM to encrypt other keys is consistent with <i>Bugiel</i>’s description of using a key that is never available outside TrEE to encrypt the session key and pre-master key. <i>See Bugiel</i> 225-227, 231.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	In addition, <i>Tugenberg</i> teaches this claim feature. For example, <i>Tugenberg</i>'s Figure 1 shows that secure memory 20 (the claimed "secure memory") includes a laser-scribed encryption key 21 (the claimed "security key"). <i>See Tugenberg</i> at 2:61-63 ("Secure memory 20 also includes laser-scribed encryption key 21 coupled to blocking gates 22."), 6:16-18 (describing "a secure memory that includes the laser-scribed encryption key"), claim 1 (reciting "a secure memory that includes a laser-scribed encryption key"), Figs. 1-3. <i>Tugenberg</i> explains "the laser-scribed encryption key is generated by laser-scribing a semiconductor die during fabrication of the secure memory to create a plurality of fixed 'ones' and 'zeroes' which make up the laser-scribed encryption key." <i>Id.</i> at claim 7; <i>see also id.</i> at claim 8 (reciting "the laser-scribed encryption key is generated burning one-time programmable fuses on a semiconductor die during fabrication of the secure memory to create a plurality of fixed 'ones' and 'zeroes' which make up the laser-scribed encryption key"), 2:16-18 ("[T]he laser-scribed encryption key is generated by laser-scribing a semiconductor die during fabrication."), 3:29-49 ("Laser-scribed encryption key 21 is a randomly generated sequence of 'ones' and 'zeros' suitable for an encryption key. The sequence is laser-scribed during fabrication of a semiconductor die . . ."). <i>Tugenberg</i> further explains that "laser-scribed encryption key 21 is comprised of a one-time programmable flash memory element, or alternatively non-volatile memory such as ROM, EEPROM, MRAM (Magnetoresistive RAM), battery backed RAM or DRAM, or other fast logic elements." <i>Id.</i> at 3:29-49. Accordingly, <i>Tugenberg</i>'s secure memory (secure memory 20) includes a security key (laser-scribed encryption key 21) that was encoded in the secure memory during a manufacturing process of the system-on-a-chip (secure processing system 14). <i>See also Tugenberg</i> at Abstract, 2:3-21, 2:59-3:18, 3:58-4:9, 4:20-31, 5:1-9, 6:12-20, 6:40-7:7, 7:17-34, 7:62-67, claims 1-2, 7-8, 10-12, 14-19. <i>Tugenberg</i> teaches using laser-scribed encryption key 21 (the claimed "security key") to encrypt a secret key / common session key. <i>See Tugenberg</i> at 4:23-27 ("[E]ach secret key [is] encrypted with laser-scribed encryption key 21 using the secure processing system 14. When needed, communication device decrypts one of the secret keys with encryption logic circuitry 23 using laser-scribed encryption key 21 . . ."), 2:12-15 ("The secure memory uses a laser-scribed encryption key coupled to encryption logic circuitry within the secure memory for encrypting and decrypting sensitive information."), 2:64-66 ("Encryption logic circuitry 23 preferably implements a symmetric encryption

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	algorithm using laser-scribed encryption key 21.”); <i>see also Tugenberg</i> at Abstract, 2:3-21, 2:59-3:18, 3:29-49, 3:58-4:9, 4:20-31, 5:1-9, 6:12-20, 6:40-7:7, 7:17-34, 7:62-67, claims 1-2, 7-8, 10-12, 14-19. <i>Tugenberg</i>’s description of using a laser-scribed encryption key that is “unique to each device” and only accessible within the secure memory, to encrypt other keys is consistent with <i>Bugiel</i>’s description of using a key that is never available outside TrEE to encrypt the session key and pre-master key. <i>See Tugenberg</i> at 2:3-21; <i>Bugiel</i> 225-227, 231. In addition, <i>Aaraj</i> teaches this claim feature by disclosing “the endorsement key (EK), is generated by the vendor or manufacturer, and stored in the nonvolatile memory within the TPM. EK should never be disclosed and is only used for . . . creating the storage root key (SRK). . . . The SRK is the root of the hierarchy of TPM keys, and is used for encrypting other migratable keys that are stored outside the TPM.” <i>Aaraj</i> at 8:6. Fig. 1 of <i>Aaraj</i> illustrates EK and SRK are stored in the secure memory within the TPM. Therefore, <i>Aaraj</i> teaches this claim feature under Daedalus’s apparent view. These disclosures are consistent with <i>Bugiel</i>’s teaching. <i>See also Aaraj</i> at 8:5, 8:30-31. Fig. 1. Modules and interfaces of a trusted platform. <i>Aaraj</i> at Fig. 1

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	In addition, <i>Gibod</i> teaches this claim feature by disclosing a device key/Master key that is “built-in” permanently in the device. <i>Gibod</i> ¶ [0051]; <i>see also</i> <i>Gibod</i> ¶ [0021], ¶ [0023], ¶ [0037], Fig. 1. <i>Gibod</i> also discloses the Master key is stored in secure memory, e.g., “secure storage.” <i>Gibod</i> ¶ [0026], claim 14. <i>Gibod</i> further discloses to use the Master key to encrypt/decrypt other keys. <i>See</i> <i>Gibod</i> ¶¶ [0009]-[0010] (“[t]o export or persist secret information, the secret information may first be encrypted using a persistent (e.g., permanent) encryption key.”), ¶ [0019] (“ . . . secret information (e.g., cryptographic keys) . . .”), ¶ [0004] (“ . . . information (e.g., cryptographic keys, . . .) . . . designed to prevent unauthorized access and/or use to secure and/or proprietary content may also need to be kept secret”), ¶ [0008] (“performing encryption/decryption operations utilizing the secret information, . . . , encrypting secret fields within a larger piece of data utilizing the secret information”), ¶ [0032], ¶ [0004], ¶ [0016], ¶¶ [0021]-[0024], ¶¶ [0026]-[0032], ¶ [0046], ¶¶ [0050]-[0051], ¶ [0053], ¶¶ [0069]-[0070], claims 11-14. <i>Tugenberg</i>’s description of using a master key that is securely embedded in a hardware security element to encrypt other keys is consistent with <i>Bugiel</i>’s description of using a key that is never available outside TrEE to encrypt the session key and pre-master key. <i>See Bugiel</i> 225-227, 231. In addition, <i>Mujtaba</i> teaches this claim feature by disclosing “Storage key 114 and/or the UID (not shown) may be hardwired in the hardware (e.g., burnt fuses) during the manufacturing of wireless processor 101.”. <i>Mujtaba</i> ¶ [0031], ¶ [0033], ¶ [0037], ¶¶ [0066]-[0068], Figs. 1 and 4. Figures 1 and 4 illustrate the storage key 114 is inside secure memory accessible only by the security engine wireless processor 101. <i>Mujtaba</i> further discloses the storage key or another key derived from the UID is used to encrypt a generated session key. <i>See</i> <i>Mujtaba</i> ¶ [0006] (“A recovery blob is generated having embedded therein the session key encrypted by the storage key. . . . Thereafter, the recovery blob . . . are sent to an authorization server.”), ¶¶ [0052]-[0053] (“a recovery blob is generated by encrypting session key 705 using storage key 114”); <i>see also</i> ¶ [0031], ¶¶ [0042]-[0044], ¶ [0055], ¶ [0057], ¶¶ [0066]-[0068], claims 10, 18, 23, Figs. 1, 4, 7-8. Under <i>Daedalus</i>’s apparent view, <i>Mujtaba</i>’s description of using a storage key or a key derived from UID that is “locally maintained” and “stored in the secure ROM” to encrypt other keys is consistent with <i>Bugiel</i>’s description of using a key that is

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	never available outside TrEE to encrypt the session key and pre-master key. <i>Mujtaba</i> ¶ [0049], ¶ [0023], ¶ [0037]; <i>see Bugiel</i> 225-227, 231. In addition, <i>Buer754</i> teaches this claim feature by disclosing “the initial KEK 130 may be programmed into the non-volatile memory 134 when the system is manufactured.” <i>Buer754</i> at 4:45-60; <i>see also Buer754</i> at 3:54-56, 5:52-62, 8:1-8, 10:34-37, 13:66-14:10, 17:5-26, 17:48-62. <i>Buer754</i> also describes the KEK is used to encrypt other keys by disclosing “these session keys are encrypted using KEK 238 and stored in a database 240 as security associations.” <i>Buer754</i>, 5:43-51; 6:36-43 (“To protect these keys, the security module 320 encrypts them with a KEK 332 before they are sent to the cryptographic accelerator 326”), 2:50-60 (“using a single KEK to encrypt all the keys associated with the network data”), Fig. 4 (encrypted packet & security association sent to cryptographic accelerator 420), Fig. 6 (step 610: encrypt and store session keys), Fig. 11 (steps 1118-1120 : encrypt and store private keys using KEK), Fig. 12 (steps 1210-1212: encrypt session keys using KEK_A and store security associations), 4:45-60 (“the initial KEK 130 may be programmed into the non-volatile memory 134 when the system is manufactured.”); <i>see also Buer754</i> at 1:42-49, 1:61-2:3, 2:9-12, 2:17-60, 3:54-56, 4:14-60, 5:43-6:12, 6:36-43, 6:53-7:23, 7:35-41, 8:1-8, 8:35-62, 9:5-12, 9:22-44, 10:23-27, 10:34-37, 10:62-11:3, 11:17-31, 11:62-12:3, 12:47-52, 13:66-14:10, 15:17-33, 16:8-15, 17:1-26, 17:43-62; Figs. 3, 5-6, 8, 11-13, claims 1-3, 6-9, 12, 14, 16. <i>Buer754</i>’s description of using a KEK to encrypt other keys is consistent with <i>Bugiel</i>’s description of using a key that is never available outside TrEE to encrypt the session key and pre-master key. <i>See Bugiel</i> 225-227, 231. In addition, at the time of the '838 patent, it was well-known to a POSITA to encode a key during manufacturing. <i>See Weise</i> ¶ [0012], ¶ [0037], ¶¶ [0042]-[0043], Abstract, claims 1, 12-13, 19; <i>Kocher</i>¹⁰ at ¶ [0007] (“a microprocessor may be manufactured having an embedded key in fuses . . . the inability to modify stored keys due to the use of one-time-programmable fuses to hold keys”); ¶ [0034] (“a secret key denoted as K_{ROOT}. . . may, for example, be stored in one or more of

¹⁰ U.S. Patent Application Publication No. 2011/0138192 to Kocher et al. (“*Kocher*”) was filed on December 2, 2010, and published on June 9, 2011. *Kocher* also claims priority to provisional application No. 61/266,948, which was filed on December 4, 2009. Thus, *Kocher* qualifies as prior art under at least pre-AIA 102(a) and 102(e).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	EEPROM, flash, fuses, or other storage on a tamper-resistant chip”); ¶ [0096] (“a secret state storage memory (702) that holds a shared cryptographic secret K_{ROOT}. The key storage memory could be implemented using a variety of techniques, such as, without limitation, fuses/antifuses, battery backed RAM, and EEPROM.”), Figs. 7-9; <i>see also Czajkowski</i>¹¹ at ¶¶ [0064]-[0065] (“a key may be programmed into an EPROM during the manufacturing of the open processor 322”); <i>Buer762</i>¹² at ¶ [0029], ¶ [0067], ¶ [0075], ¶ [0129], ¶ [0155], ¶ [0157]; <i>Azema</i>¹³ at 3 (“Unique 128-bit hardware key”), 4 (M-Shield mobile security system solution infrastructure); <i>Davis</i> at 5:24-29 (“The non-volatile memory element 610 contains at least a shared secret key, which is also imprinted into the secret key storage element 420 normally during manufacture when the cryptographic unit 335 and the chipset are powered up and in communication with each other.”), Figs. 3-4. Exemplary disclosures from the prior art include the following: <i>Bugiel</i> discloses the following at 230-231: A detailed description of the ObC architecture can be found in [7]. In a nutshell, it provides a bytecode interpreter residing in the TrEE, which can execute lightweight bytecode, compiled from scripts written in assembly code. Moreover, the interpreter provides an interface for commonly used cryptographic primitives. Furthermore, the

¹¹ U.S. Patent Application Publication No. 2010/0191959 to Czajkowski (“*Czajkowski*”) was filed on January 7, 2010 and published July 29, 2010. Thus, *Czajkowski* qualifies as prior art under at least pre-AIA 102(a), (b) and (e).

¹² U.S. Patent Application Publication No. 2006/0072762 to Buer (“*Buer762*”) was filed on June 21, 2005 and published April 6, 2006. Thus, *Buer762* qualifies as prior art under at least pre-AIA 102(a), (b) and (e).

¹³ J. Azema, et al., “M-Shield™ Mobile Security Technology: making wireless secure,” (“*Azema*”) (2005). *Azema* was cited in numerous publications before the earliest priority date of the '838 patent, including e.g., J. Kaavi, “Strong authentication with mobile phones,” Aalto University, T-110.5290 Seminar on Networking Security (Fall 2010); M. Richarme, “The Virtual SIM: a feasibility study,” master thesis (April 21, 2008); B. Parno, et al., “Help me help you: using trustworthy host-based information in the network” (November 18, 2009). Thus, *Azema* qualifies as prior art under at least pre-AIA 102(a), (b).

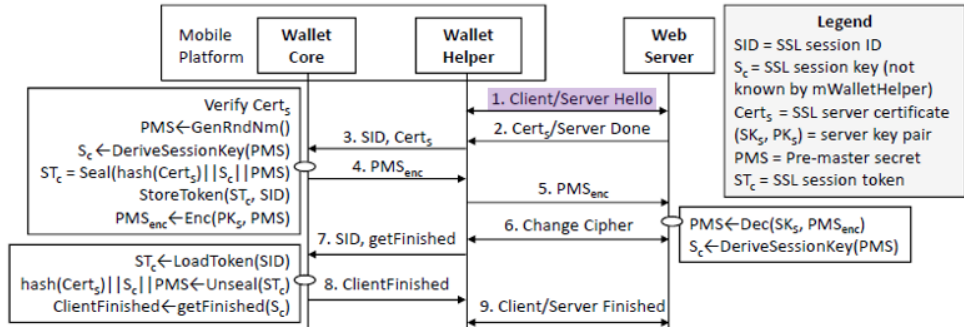
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	provisioning subsystem of ObC enables in our implementation the out-of-band provisioning of users' passwords. The sealing functionality provided by the interpreter is used to implement SecureStorage. When sealed, data is encrypted under a key that is never available outside the TrEE. Further, this key is cryptographically bound to scripts, thus isolating the SecureStorage from other, unauthorized (and potentially malicious) scripts. <i>See also Bugiel</i> at 225-231, 235-236, n. 8. <i>See also Chou</i> Abstract, ¶ [0003], ¶ [0005], ¶¶ [0007]-[0009], ¶¶ [0017]-[0019], ¶ [0021], ¶¶ [0023]-[0028], ¶¶ [0034]-[0041], ¶¶ [0054]-[0064], ¶¶ [0069]-[0071], claims 1, 4, 6, 8, 19, Figs. 1 and 3. <i>See also Tkacik</i> ¶¶ [0002]-[0004], ¶¶ [0006]-[0009], ¶ [0019], ¶ [0029], ¶ [0039], ¶¶ [0044]-[0046], ¶¶ [0048]-[0054], ¶ [0057], ¶¶ [0069]-[0070], ¶ [0074], ¶¶ [0079]-[0080], ¶¶ [0083]-[0085], ¶¶ [0088]-[0089], Figs. 2-8, Abstract. <i>See also Paaske</i> at ¶¶ [0043]-[0044], ¶ [0056], ¶¶ [0059]-[0060], ¶ [0063], ¶ [0101], ¶ [0108], ¶¶ [0110]-[0111], ¶¶ [0217]-[0227], Figs. 1-3 <i>See also Tugenberg</i> at Abstract, 2:3-21, 2:59-3:18, 3:29-49, 3:58-4:9, 4:20-31, 5:1-9, 6:12-20, 6:40-7:7, 7:17-34, 7:62-67, claims 1-2, 7-8, 10-12, 14-19, Fig. 1-3. <i>See also Aaraj</i> at 8:5-6, 8:30-31. <i>See also Gibod</i> ¶ [0004], ¶¶ [0008]-[0010], ¶ [0016], ¶ [0019], ¶¶ [0021]-[0024], ¶¶ [0026]-[0032], ¶ [0037], ¶ [0046], ¶¶ [0050]-[0051], ¶ [0053], ¶¶ [0069]-[0070], claims 11-14, Fig. 1. <i>See also Mujtaba</i> ¶ [0006], ¶ [0023], ¶ [0031], ¶ [0033], ¶ [0037], ¶¶ [0042]-[0044], ¶ [0049], ¶¶ [0052]-[0053], ¶ [0055], ¶ [0057], ¶¶ [0066]-[0068], claims 10, 18, 23, Figs. 1, 4, 7-8. <i>See also Buer754</i> at 1:42-49, 1:61-2:3, 2:9-12, 2:17-60, 3:42-48, 3:54-56, 4:14-60, 5:43-6:12, 6:36-43, 6:53-7:23, 7:35-41, 8:1-8, 8:35-62, 9:5-12, 9:22-44, 10:23-27, 10:34-37, 10:62-11:3, 11:17-31, 11:62-

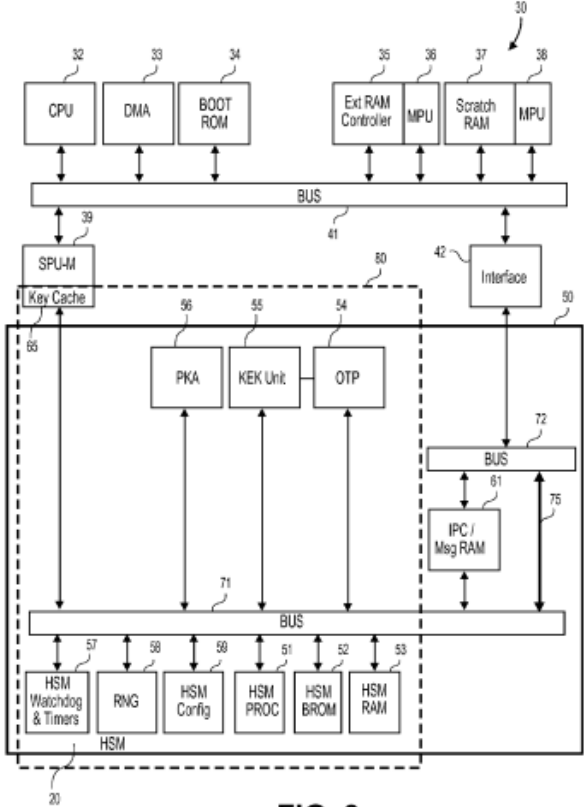
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	12:3, 12:47-52, 13:66-14:10, 15:17-33, 16:8-15, 17:1-26, 17:43-62; Figs. 3, 5-6, 8, 11-13, claims 1-3, 6-9, 12, 14, 16. <i>See supra</i> [1pre], [1a-2].
[1b] the security engine to: generate a random nonce for initiating a request for a secure communication session with a remote server over a network using the nonce;	The grounds alleged at the beginning of this chart disclose or suggest “the security engine to: generate a random nonce for initiating a request for a secure communication session with a remote server over a network using the nonce.” <i>Bugiel</i> teaches the claimed “secure communication session with a remote server over a network.” <i>Bugiel</i> explains: “For secure communication between the device and the remote web server we utilize SSL/TLS protocol.” <i>Bugiel</i> at 222; <i>see also generally Bugiel</i> at 219-236 (describing mobile communication based on known security protocols such as SSL). This description matches the '838 patent's description in its Background section. <i>See</i> '838 patent at 1:10-24 (“One commonly used cryptographic communication protocol is the Secure Sockets Layer (SSL) protocol. The SSL protocol, and the related Transport Layer Security (TLS) protocol, is used for many different types of secure communication sessions including, for example, secure web-browsing, ecommerce, secure upgrades, and other secure communication sessions between two computing devices on an untrusted network such as the Internet . . .”). In addition, <i>Bugiel</i>'s Figure 2 shows the mobile device exchanges ClientHello and ServerHello messages with the Web Server as the first step of the SSL/TLS handshake. <i>Bugiel</i> thus also teaches the claimed “initiating a request for a secure communication session with a remote server over a network.” A POSITA would have understood that a ClientHello message sent by a client mobile device includes a random number generated by the client (the claimed “generate a random nonce”) in the SSL/TLS protocols, e.g., the <i>TLS-1.2</i> protocol incorporated by reference in <i>Bugiel</i>. <i>See Bugiel</i> at 225-226, 236, Fig. 2. For example, as <i>Chou</i> explains, a “generated random value is typically used as a seed for a cryptographic algorithm.” <i>Chou</i> ¶ [0042].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	 Fig. 2. SSL/TLS handshake <i>Bugiel</i> at Fig. 2 (annotated). The combinations alleged here use one of <i>Chou</i>, <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i>'s SoC to implement <i>Bugiel</i>'s communications method, including to implement IKE or SSL/TLS Handshake Protocols, to establish a secure communication. For example, <i>Chou</i>'s Figure 3 shows that HSM 20 (part of the claimed "security engine") includes RNG (random number generator) 58, which generates random value as a seed for cryptographic algorithms. <i>Chou</i> ¶ [0034], ¶ [0042]. <i>Chou</i> also teaches using a number of cryptographic algorithms, such as Diffie-Hellman key exchange algorithm and RSA key generation. <i>See id.</i> ¶ [0034], ¶¶ [0043]-[0053]. A POSITA would have understood, for example, that RNG 58 generates random number as a seed for the key exchange process, along with other parameters to generate session keys. A POSITA also would have understood such key exchange algorithms are used to establish secure communication with a remote server over a network using the nonce. For example, <i>Chou</i> explains that its invention is used for "secure data transfer . . . to perform secure financial transactions over a network, such as the Internet." <i>Chou</i> ¶¶ [0005]-[0006], Figs. 4-5 (describing wireless communication with remote server). <i>see also Chou</i> ¶¶ [0005]-[0007], ¶ [0015], ¶ [0017], ¶ [0020], ¶¶ [0022]-[0023], ¶ [0034], ¶ [0037],

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	¶¶ [0041]-[0053], ¶ [0063], ¶ [0066], ¶ [0068], ¶¶ [0070]-[0071], Abstract, claims 1, 4, 6, 8, 11-12, 15-16, 19, Figs. 3-5. A POSITA thus would have found it obvious to use <i>Chou</i>'s RNG 58 to “generate a random nonce,” as claimed, for use in the ClientHello message, which <i>Bugiel</i>'s TruWalletM method describes to request a secure IKE or SSL/TLS communication session with a remote network server.  FIG. 3 <i>Chou</i>, Fig. 3.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	In addition to <i>Chou</i> and <i>Bugiel</i>, at the time of the '838 patent, it was well-known to a POSITA to generating a random nonce for initiating a request for secure communication with a remote server as part of the SSL/TLS or IKE protocol. For example, <i>Buer754</i> discloses initiator generates random number x_i and creates public number y_i based on x_i to send to responder along with a nonce N_i. See <i>Buer754</i> at 13:1-44, 1:6-42 (SSL or IPsec provides secure Internet transmission between a client and server), 1:61-2:13 (security module for protecting keys generated in SSL or IPsec), 6:13-31 (“Fig. 3 shows a security module 320 uses the SSL protocol to provide data security in a network”); see also <i>Buer754</i> at 2:17-24, 2:50-60, 3:42-7:64, 8:35-56, 8:63-9:47, 10:14-56, 11:3-39, 11:62-12:23, 12:62-67, 13:21-16:19, 16:26-17:62, Abstract, Figs 1-3, 6, 10-13, claims 1, 3-6, 8-20; see also <i>Tugenberg</i> at 5:48-61 (“[t]he host processor uses [a] certificate to establish a secure session with, for example, an internet vendor. Secure Socket Layer (SSL), for example, is a standard technique used in the internet community to establish such a session.”), Title, Abstract, 4:54-58, 5:55-61, 1:7-10, 1:26-36, 1:48-55, 2:31-58, 4:20-31, 4:38-49, 4:54-58, 5:10-15, 6:12-20, 6:26-39, 8:7-20; see also <i>Aaraj</i> at 8:6 (“TPM’s random number generator (RNG), key generator and RSA engine, support the generation of RSA and symmetric keys, and RSA cryptographic operations, such as signing and encryption/decryption.”), 8:9 (SSL), 8:2, 8:5, 8:11-16, 8:30-8:31; see also <i>Tkacik</i> ¶¶ [0006]-[0009], ¶ [0021], ¶¶ [0047]-[0048], ¶ [0059], ¶¶ [0078]-[0082], ¶¶ [0084]-[0087], ¶ [0089], Figs. 2, 5, 8; see also <i>Paaske</i> ¶ [0154] (random number generator), ¶ [0191], ¶ [0032], ¶ [0006], Fig. 2; see also <i>Gibod</i> at ¶ [0052] (random number generation), ¶ [0071] (TLS), ¶ [0016], ¶¶ [0069]-[0070]. Additionally, each of the SSL/TLS protocols teaches this claim feature. For example, <i>Bugiel</i> incorporates <i>TLS-1.2</i> by reference. See <i>Bugiel</i> at 225, 236; see also <i>Bugiel</i> at 223-227. <i>TLS-1.2</i> explains that “TLS should be used to establish a secure connection between parties.” <i>TLS-1.2</i> at 6. The first step of the TLS Handshake Protocol is for the client to generate ClientHello.random (the claimed “generate a random nonce”) and send a client hello message including ClientHello.random to the server (the claimed “initiating a request for a secure communication session with a remote server over a network using the nonce”). See <i>id.</i> at 34 (“[T]wo random values are generated and exchanged: ClientHello.random and ServerHello.random”), 36-37 (Figures 1-2 showing initiating a TLS handshake process by sending a ClientHello message), 39 (“When a client first connects to a server, it is required to send the ClientHello as its first message. . . The ClientHello message includes a random structure.”); see also <i>TLS-1.2</i> at 4, 6, 17-18, 26, 33-43, 52, 64, 71, 73, 77, 85, 90, 92, 94, 100. <i>SSL-3.0</i>,

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>TLS-1.0</i>, <i>TLS-1.1</i> also provide similar disclosures. <i>See TLS-1.1</i> at 32 (“two random values are generated and exchanged: ClientHello.random and ServerHello.random”), 33-34, 36, 5, 16, 24, 31, 36-39, 44, 52-53, 57-58, 64, 70, 73, 75-76, 78, 83; <i>TLS-1.0</i> at 30 (“two random values are generated and exchanged: ClientHello.random and ServerHello.random”), 31-32, 34, 4, 14-15, 21-23, 29, 35-37, 41, 47, 51-52, 56, 64, 67-68, 70-72; <i>SSL-3.0</i> at 18 (“two random values are generated and exchanged: ClientHello.random and ServerHello.random”), 19-21, 4, 11, 22-24, 26, 32, 34-35, 38, 49, 52-53, 56-59. Each TLS/SSL protocol explains that that ClientHello.random and ServerHello.random are random nonces exchanged between client and remote server to avoid “replay” attack in SSL/TLS protocols <i>See TLS-1.2</i> at 91-92; <i>SSL-3.0</i> at 55, 57; <i>TLS-1.0</i> at 69-71; <i>TLS-1.1</i> at 74-76.¹⁴ In addition, <i>TLS-1.0</i> incorporates by reference the original SSL patent U.S. Patent No. 5,657,390 by T. Elgamal et al. (“<i>Elgamal</i>”), which was filed on August 25, 1995 and patented on August 12, 1997. <i>See TLS-1.0</i> at 74.

¹⁴ Before the earliest possible priority date of the '838 patent, many patent documents describe the random numbers exchanged between the server and client as random nonce. *See* U.S. Patent No. 6,643,774 to McGarvey (e.g., at 7:16-8:3), which was filed on April 8, 1999 and patented on November 4, 2003; U.S. Patent No. 7,039,946 to Binding et al. (e.g., at 10:41-57, 11:52-12:4, 14:52-15:6), which was filed on October 12, 1999 and patented on May 2, 2006; U.S. Patent No. 6,775,772 to Binding et al. (e.g., at 9:64-10:13, 11:5-23, 14:1-21), which was filed on October 12, 1999 and patented on August 10, 2004; U.S. Patent No. 6,694,431 to Binding et al. (e.g., at 10:10-26, 11:18-37, 14:15-35), which was filed on October 12, 1999 and patented on February 17, 2004; U.S. Patent No. 6,751,731 to Binding et al. (e.g., 9:63-10:12, 11:12-18, 14:15-35), which was filed on October 12, 1999 and patented on June 15, 2004; U.S. Patent Application Publication No. 2010/0205443 to Zhao et al. (e.g., at ¶ [0114], ¶¶ [0175]-[0177], ¶ [0186]), which was filed on April 23, 2010 and published on August 12, 2010; U.S. Patent Application Publication No. 2011/0231653 to Bollay et al., (e.g., at ¶¶ [0003]-[0005], ¶¶ [0022]-[0029], ¶ [0036], ¶ [0062], ¶ [0070], ¶ [0090], ¶ [0137]), which was filed on December 13, 2010 and published on September 22, 2011; U.S. Patent No. 5,657,390 to Elgamal et al (e.g., at 6:35-55, 33:24-37, Figs. 9-11), which was filed on August 25, 1995 and patented on August 12, 1997; U.S. Patent Application Publication No. 2010/0042824 to Lee et al. (e.g., at ¶ [0074], ¶ [0081], ¶¶ [0087]-[0088], ¶ [0118], ¶ [0130]), which was filed on August 14, 2009 and published on February 18, 2010; *Buer-754* at 1:36-41, 1:61-2:8, 6:13-31, 13:21-37, 13:45-65, 14:11-34, 14:43-55; EP1706956B1 to Cam-Winget et al. (e.g., at ¶¶ [0139]-[0142]), which was filed on January 10, 2005 and published on October , 2006; U.S. Patent Application Publication No. 2004/0005061 to Buer et al. (e.g., at ¶ [0005], ¶¶ [0009]-[0010], ¶¶ [0050]-[0051], ¶¶ [0108]-[0115], ¶¶ [0117]-[0124], ¶¶ [0127]-[0134], ¶¶ [0136]-[0140]), which was filed on July 8, 2002 and published on January 8, 2004.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>Elgamal</i> also teaches including “a random number used to ensure channel integrity” in the client-hello message, which the server responds with a server-hello message including a server random number. <i>Elgamal</i> at 7:13-40; <i>see also id.</i> at Figs. 4-7. <i>Elgamal</i> further describes the random number as “nonce” because they are “used to defeat ‘playback’ attacks.” <i>Elgamal</i> at 33:24-31. Additionally, the combinations based on one of the IPsec IKE protocols (<i>RFC-4306</i> or <i>RFC-5996</i>) also discloses or suggests generating a random nonce for initiating a request for a secure communication session with a remote server over a network using the nonce. For example, <i>RFC-5996</i> teaches secure communication session with a remote server over a network by describing “IKE performs mutual authentication between two parties and establishes an IKE security association (SA) that includes shared secret information that can be used to efficiently establish SAs for Encapsulating Security Payload (ESP) [ESP] or Authentication Header (AH) [AH] and a set of cryptographic algorithms to be used by the SAs to protect the traffic that they carry.” <i>RFC-5996</i> at 5; <i>see also RFC-5996</i> at 1 (“IKE is a component of IPsec used for performing mutual authentication and establishing and maintaining Security Associations (SAs).”), 5-9, 40, 52, 54, 64-65. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 3 (“IKE performs mutual authentication between two parties and establishes and IKE security association (SA) that includes shared secret information that can be used to efficiently establish SAs for Encapsulating Security Payload (ESP) [RFC4303] and/or Authentication Header (AH) [RFC4302] and a set of cryptographic algorithms to be used by the SAs to protect the traffic that they carry.”), 1, 3-7, 13, 21, 33-34. <i>RFC-5996</i> describes that “[c]ommunication using IKE always begins with IKE_SA_INIT and IKE_AUTH exchange (known in IKEv1 as Phase 1). These initial exchanges normally consist of four messages . . .” <i>RFC-5996</i> at 9. After the initial exchange, a CREATE_CHILD_SA exchange (or Phase 2 exchange) “is used to create new Child SAs and to rekey both IKE SAs and Child SAs.” <i>RFC-5996</i> at 13; <i>see also RFC-5996</i> at 5-6, 9-18, 20, 30-33, 44, 52-53, 132, 134-137. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 7 (“[c]ommunication using IKE always begins with IKE_SA_INIT and IKE_AUTH exchange (known in IKEv1 as Phase 1). These initial exchanges

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	normally consist of four messages . . .”), 9-11 (describing CREATE_CHILD_SA exchange in phase 2); <i>see also RFC-4306</i> at 4, 7-12, 19-20, 22-23, 26, 33-34, 86, 96. <i>RFC-5996</i> discloses in both phases of the IKE, the initiator generates and sends a random nonce Ni to the responder to initiate a request for a secure communication session with a remote server over a network using the nonce: <pre style="margin-left: 40px;"> The initial exchanges are as follows: Initiator Responder ----- HDR, SAi1, KEi, Ni --> HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. The SAi1 payload states the cryptographic algorithms the initiator supports for the IKE SA. The KE payload sends the initiator's Diffie-Hellman value. Ni is the initiator's nonce. <-- HDR, SAR1, KEr, Nr, [CERTREQ]</pre> <i>RFC-5996</i> at 10; <i>see also RFC-5996</i> at 6, 9-11, 13-16, 33, 37-39, 44, 46-48, 50, 52-53, 62, 69, 74, 96, 122, 134, 137. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 4 (“The first request/response of an IKE session (IKE_SA_INIT) negotiates security parameters for the IKE_SA, sends nonces, and sends Diffie-Hellman values.”), 7-8, 10, 19, 23, 26, 28-30, 32-34, 39, 45, 64, 69. Therefore, the grounds alleged here disclose or suggest generating a random nonce for initiating a request for a secure communication session with a remote server over a network using the nonce. Exemplary disclosures in the prior art include the following: <i>Bugiel</i> discloses the following at 225-227 (4.1 Establishing SSL/TLS connection and managing two logical SSL/TLS subchannels):

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

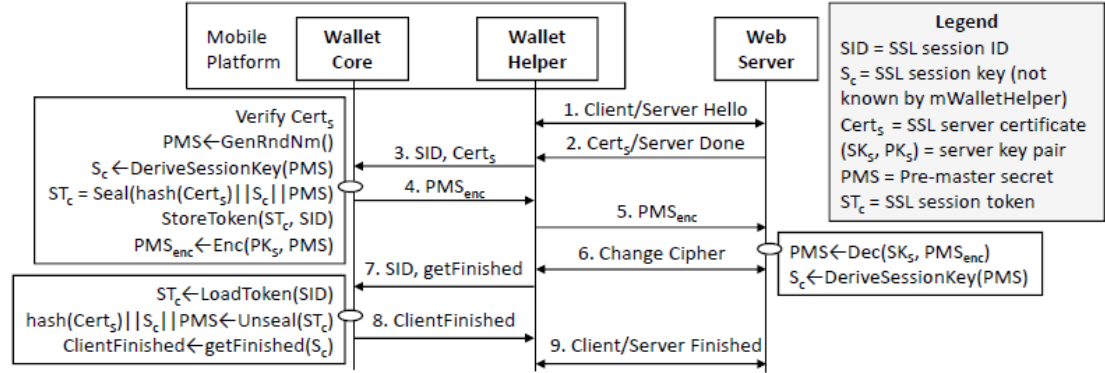
The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	Figure 2 illustrates the SSL handshake protocol. The client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters. In the following steps the server's certificate $Cert_s$ is sent and verified, the SSL/TLS session key S_c is generated (which is never disclosed to WalletHelper) and sealed (together with pre-master secret PMS and the hash value $\text{hash}(Cert_s)$). The resulting session-token ST_c is stored in a secure storage and can be loaded and unsealed later using the session identifier SID. ... All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16].  The diagram illustrates the SSL/TLS handshake protocol involving four entities: Mobile Platform, Wallet Core, Wallet Helper, and Web Server. The process is as follows: Mobile Platform: - Verify $Cert_s$ $PMS \leftarrow \text{GenRndNm}()$ $S_c \leftarrow \text{DeriveSessionKey}(PMS)$ $ST_c = \text{Seal}(\text{hash}(Cert_s) S_c PMS)$ - StoreToken(ST_c, SID) $PMS_{enc} \leftarrow \text{Enc}(PK_s, PMS)$ - Later: $ST_c \leftarrow \text{LoadToken}(SID)$ $\text{hash}(Cert_s) S_c PMS \leftarrow \text{Unseal}(ST_c)$ - ClientFinished $\leftarrow \text{getFinished}(S_c)$ Wallet Core: - Receives 3. SID, $Cert_s$ from Wallet Helper - Sends 4. PMS_{enc} to Wallet Helper - Receives 7. SID, getFinished from Web Server - Sends 8. ClientFinished to Web Server Wallet Helper: - Sends 1. Client/Server Hello to Web Server - Receives 2. $Cert_s$/Server Done from Web Server - Receives 5. PMS_{enc} from Web Server - Sends 6. Change Cipher to Web Server - Receives 9. Client/Server Finished from Web Server Web Server: - Receives 3. SID, $Cert_s$ from Wallet Helper - Sends 4. PMS_{enc} to Wallet Helper - Receives 6. Change Cipher from Wallet Helper - Performs: $PMS \leftarrow \text{Dec}(SK_s, PMS_{enc})$ - Performs: $S_c \leftarrow \text{DeriveSessionKey}(PMS)$ Legend: - SID = SSL session ID S_c = SSL session key (not known by mWalletHelper) $Cert_s$ = SSL server certificate (SK_s, PK_s) = server key pair - PMS = Pre-master secret ST_c = SSL session token

Fig. 2. SSL/TLS handshake

See also *Bugiel* at 219-236.

Chou discloses the following at ¶¶ [0042]-[0053]:

RNG 58 uses a random bit generator to generate true random numbers. In one embodiment, RNG 58 utilizes free running oscillators to capture thermal noise as the source of

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	randomness and is controlled via configuration registers and generates one bit at a time into a 512 bit capture FIFO (First-In-First-Out). The generated random value is typically used as a seed for a cryptographic algorithm and the value from RNG 58 is typically post processed and not used directly. In one embodiment, the random bit rate is a nominal 1 MHz generation rate providing 1 Mbps of random data. PKA module 56 is a specialized processor that provides acceleration for a number of algorithms. In one embodiment, PKA module 56 is an accelerator that is operable for the following algorithms: 1. Diffie-Hellman public value and shared key computation of modulus sizes up to 4096 bits. 2. 1024-bit DSA signature with SHA-1 hash or 2048-bit DSA signature with SHA-256 hash. 3. RSA encryption of modulus sizes up to 4096-bit key size. 4. RSA decryption of modulus sizes up to 4096-bits using Chinese Remainder Theorem. 5. RSA key generation with Rabin-Miller number selection. 6. Elliptic curve Diffie-Hellman in prime field GF(p). ... It is to be noted that these algorithms are listed for exemplary purpose and that other embodiments of PKA module 56 may support a different list of algorithms, including algorithms not listed above. <i>See also Chou ¶¶ [0005]-[0007], ¶ [0015], ¶ [0017], ¶ [0020], ¶¶ [0022]-[0023], ¶ [0034], ¶ [0037], ¶¶ [0041]-[0063], ¶ [0066], ¶ [0068], ¶¶ [0070]-[0071], Abstract, claims 1, 4, 6, 8, 11-12, 15-16, 19, Figs. 3-5.</i>

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also TLS-1.2</i> at 3-6, 17-18, 23, 26, 33-43, 52, 64, 71, 73, 77, 85, 90-92, 94, 100. <i>See also TLS-1.1</i> at 5, 16, 24, 31-34, 36-39, 44, 52-53, 57-58, 64, 70, 73-76, 78, 83. <i>See also TLS-1.0</i> at 4, 14-15, 21-23, 29-32, 34-37, 41, 47, 51-52, 56, 64, 67-72, 74. <i>See also SSL-3.0</i> at 4, 11, 18-24, 26, 32, 34-35, 38, 49, 52-53, 55-59. <i>See also RFC-5996</i> at 1, 5-18, 20, 30-33, 37-40, 44, 46-48, 50, 52-54, 62, 64-65, 69, 74, 96, 122, 132, 134-137. <i>See RFC-4306</i> at 1, 3-13, 19-23, 26, 28-30, 32-34, 39, 45, 64, 69, 86, 96. <i>see also Buer754</i> at 1:36-42, 1:61-2:13, 2:17-24, 2:50-60, 3:42-7:64, 6:13-31, 8:35-56, 8:63-9:47, 10:14-56, 11:3-39, 11:62-12:23, 12:4-23, 12:62-67, 13:1-16:19, 16:26-17:62, Abstract, Figs 1-3, 6, 10-13, claims 1, 3-6, 8-20. <i>See also Tugenberg</i> at Title, Abstract, 4:54-58, 5:48-61, 1:7-10, 1:26-36, 1:48-55, 2:31-58, 4:20-31, 4:38-49, 4:54-58, 5:10-15, 6:8-20, 6:26-39, 8:7-20, Figs. 1 and 3, claims 1, 5, 12. <i>See also Aaraj</i> at 8:2, 8:5-6, 8:9, 8:11-16, 8:30-8:31. <i>See also Tkacik</i> ¶¶ [0047]-[0048] (random number generator), ¶¶ [0006]-[0009], ¶ [0021], ¶ [0059], ¶¶ [0078]-[0082], ¶¶ [0084]-[0089], Figs. 1-2, 5-8, claims 1, 9-10, 16-17. <i>See also Paaske</i> ¶ [0003], ¶ [0006], ¶ [0027], ¶ [0032], ¶ [0154], ¶ [0191], ¶¶ [0216]-[0227], Fig. 2. <i>See also Gibod</i> at ¶ [0016], ¶ [0052], ¶ [0071], ¶ [0066], ¶¶ [0068]-[0070], Fig. 5. <i>See also</i> disclosures of [1a-1].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
[1c] perform a cryptographic key exchange with the remote server;	The grounds alleged at the beginning of this chart disclose or suggest the security engine to “perform a cryptographic key exchange with the remote server.” <i>Bugiel</i> explains: “Figure 2 illustrates the SSL handshake protocol. The client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters. In the following steps the server’s certificate Cert_s is sent and verified, the SSL/TLS session key S_c is generated . . . All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16].” <i>Bugiel</i>, at 225-226. <i>Bugiel</i> thus specifically incorporates by reference document “[16],” which is <i>TLS-1.2</i>, for more details of implementing the cryptographic key exchange messages during a TLS/SSL handshake (as well as other standard secure communication protocols, such as IPsec IKE, etc.) with a remote server (WebServer). See <i>Bugiel</i> at 225-226, 236; see also generally <i>Bugiel</i> on SSL/TLS and alternative security protocols at 220-236. <i>Bugiel</i>’s Figure 2 also illustrates SSL/TLS handshake, which involves performing the cryptographic key exchange with WebServer (the claimed “remote server”), where the “WalletHelper acts as an SSL/TLS proxy.” <i>Bugiel</i> at 226. <pre> sequenceDiagram participant MP as Mobile Platform participant WC as Wallet Core participant WH as Wallet Helper participant WS as Web Server Note over MP: Verify Cert_s PMS ← GenRndNm() Note over MP: S_c ← DeriveSessionKey(PMS) Note over MP: ST_c = Seal(hash(Cert_s) S_c PMS) Note over MP: StoreToken(ST_c, SID) Note over MP: PMS_enc ← Enc(PK_s, PMS) WH->>WS: 1. Client/Server Hello WS->>WH: 2. Cert_s/Server Done WH->>WC: 3. SID, Cert_s WC->>WH: 4. PMS_enc WH->>WS: 5. PMS_enc WS->>WH: 6. Change Cipher WH->>WC: 7. SID, getFinished Note over MP: ST_c ← LoadToken(SID) Note over MP: hash(Cert_s) S_c PMS ← Unseal(ST_c) Note over MP: ClientFinished ← getFinished(S_c) WH->>WS: 8. ClientFinished WS->>WH: 9. Client/Server Finished Note over WS: Legend SID = SSL session ID S_c = SSL session key (not known by mWalletHelper) Cert_s = SSL server certificate (SK_s, PK_s) = server key pair PMS = Pre-master secret ST_c = SSL session token Note over WS: PMS ← Dec(SK_s, PMS_enc) S_c ← DeriveSessionKey(PMS) </pre> Fig. 2. SSL/TLS handshake <i>Bugiel</i> at Fig. 2.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	The configuration of <i>Bugiel</i>'s Figure 2, in which the WalletCore communicates with the WebServer using the WalletHelper as a proxy, is substantially similar to Fig. 3 of the '838 patent, in which a security engine driver of a client device communicates with a server with a client secure communication application 304 as a proxy. <i>See</i> '838 pat. at 6:26-33 ("To establish the secure communication session, a security engine driver 302 of the security engine 110 communicates with a firmware of the security engine 110 and a client secure communication application 304 executed on the client device 102. The client secure communication application 304 also communicates with a corresponding server secure communication application 306 executed on the server 104.") FIG. 3 '838 pat. at Fig. 3.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	In addition, <i>Chou</i> teaches performing cryptographic key exchange by explaining that HSM 20 includes a Public Key Accelerator (PKA) module 56 to perform a plurality of key algorithms, such as Diffie-Hellman shared secret computation, RSA key generation, etc. <i>Chou</i> ¶¶ [0043]-[0053]. This is consistent with the '838 patent's description that "[t]he cryptographic key exchange may be, for example, a Rivest-Shamir-Adleman (RSA) key exchange or a Diffie-Hellman key exchange." '838 pat. at Abstract; <i>see also id.</i> at 3:28-33 ("To establish the secure communication session, the client device 102 and the server 104 perform a cryptographic key exchange, such as a Diffie-Hellman or a Rivest-Shamir-Adleman (RSA) key exchange, to generate a secret symmetric session key.") <i>Chou</i> also explains that SPU-M 39 "operates as a hardware accelerator for symmetric key algorithm." <i>Chou</i> ¶¶ [0054]-[0055]. A POSITA would have recognized that this teaching indicates the keys generated based on the key exchange algorithms are symmetric keys. <i>See also Chou</i> ¶ [0005], ¶ [0035], ¶ [0056], ¶¶ [0061]-[0062], ¶¶ [0069]-[0070], Figs. 1, 3-5.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	The diagram illustrates a system architecture. At the top, a CPU (32), DMA (33), and BOOT ROM (34) are connected to a central BUS (41). Below the BUS, there is an SPU-M (39) and a Key Cache (40). To the right, an Ext RAM Controller (35), MPU (36), Scratch RAM (37), and another MPU (38) are also connected to the BUS. A dashed box labeled 20 encloses a set of components including a PKA (56), KEK Unit (55), and OTP (54). Below these, there is an HSM (57) containing a Watchdog & Timers (57), RNG (58), HSM Config (59), HSM PROC (60), HSM BROM (61), and HSM RAM (62). An interface (42) connects the system to an external BUS (72), which is further connected to an IPC/Msg RAM (61) and another BUS (75). FIG. 3 <i>Chou</i>, Fig. 3. Further, <i>TLS-1.2</i> explains that the TLS version 1.2 Handshake Protocol involves exchanging cryptographic keys between a client and server to generate a master secret. <i>TLS-1.2</i> at 33-37 (Handshake Protocol overview), 37-64 (Handshake Protocol), at 64-65 (Computing the Master Secret), Figs. 1-2. The master secret is then used to generate encryption keys. <i>Id.</i> at 80. <i>TLS-1.2</i> further explains a master secret is associated with a secure session, and “[t]he general goal of the key exchange process is to create a pre_master_secret . . . to generate the master_secret The master_secret is required to generate . . . encryption keys” <i>Id.</i> at 91; <i>see also TLS-1.2</i> at 4-6, 23, 29, 33-65, 70-76, 78-83, 86-87, 91-94, 96-100.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<pre> Client Server ClientHello -----> ServerHello Certificate* ServerKeyExchange* CertificateRequest* ServerHelloDone <----- Certificate* ClientKeyExchange CertificateVerify* [ChangeCipherSpec] Finished -----> [ChangeCipherSpec] Finished <----- Application Data <-----> Application Data </pre> Figure 1. Message flow for a full handshake <i>TLS-1.2</i> at Figure 1. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provide similar disclosures as <i>TLS-1.2</i>. See <i>SSL-3.0</i> at 18-20 (Handshake Protocol Overview), 20-32 (Handshake Protocol), 32 (computing the master secret), 33 (RSA and Diffie-Hellman key exchange algorithms), 33-35 (Converting the master secret into keys and MAC secrets), 45 (master secret is “Secure secret data used for generating encryption keys, ...”), 55 (“[t]he general goal of the key exchange process is to create a pre_master_secret The pre_master_secret will be used to generate the master_secret (see Section 6.1). The master_secret is required to generate . . . encryption keys”); see also <i>SSL-3.0</i> at 4, 10, 16-35, 37-42, 45-49, 51-52, 55-62; see <i>TLS-1.0</i> at 29-32 (Handshake Protocol Overview), 32-47 (Handshake Protocol), 47 (the master secret), 48 (RSA and Diffie-Hellman key exchange algorithms), 59 (master secret is “Secure secret data used for generating encryption keys, ...”), 69 (“[t]he general goal of the key exchange process is to create a pre_master_secret The pre_master_secret will be used to generate the master_secret (see Section 8.1). The master_secret is required to generate . . . encryption keys”), Figs. 1-2; see also <i>TLS-1.0</i> at 4, 20, 22, 25, 27-48, 51-54, 59-62, 64, 66-74, 76; <i>TLS-1.1</i> at 31-34 (Handshake Protocol Overview), 34-52 (Handshake Protocol), 52-53 (the master secret), 53 (RSA and Diffie-Hellman key exchange algorithms), 66 (master secret is “Secure secret data used for generating encryption keys, ...”), 74-75 (“[t]he general goal of the key exchange process is to create a pre_master_secret The pre_master_secret will be used to generate the master_secret (see

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>				
	Section 8.1). The master_secret is required to generate . . . encryption keys . . .”), Figs. 1-2; <i>see also TLS-1.1</i> at 4-5, 22, 27, 30-53, 57-60, 66-68, 72, 74-77, 80-83. Additionally, in the combinations based on IPsec IKE protocols (<i>RFC-5996</i> or <i>RFC-4306</i>), each of the IPsec IKE protocol teaches the claimed “perform a cryptographic key exchange with the remote server.” For example, <i>RFC-5996</i> discloses “[t]his document describes version 2 of the Internet Key Exchange (IKE) protocol. IKE is a component of IPsec used for performing mutual authentication and establishing and maintaining security associations (SAs).” <i>RFC-5996</i> at 1. <i>RFC-5996</i> further describes “[c]ommunication using IKE always begins with IKE_SA_INIT and IKE_AUTH exchanges (known in IKEv1 as Phase 1). These initial exchanges normally consist of four messages All communications using IKE consist of request/response pairs. . . . The first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange [DH].” <i>RFC-5996</i> at 9. <i>RFC-5996</i> then continues to describe the IKE_AUTH exchanges, CREATE_CHILD_SA Exchange and INFORMATIONAL Exchange. <i>See RFC-5996</i> at 9-19; <i>see also RFC-5996</i> at 3, 5-6, 9-23, 25-26, 31-33, 39-41, 44-48, 50-54, 72, 74, 80-83, 86-87, 119-121, 132-137. The initial exchanges are as follows: <table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; width: 50%;">Initiator</th><th style="text-align: right; width: 50%;">Responder</th></tr> </thead> <tbody> <tr> <td colspan="2" style="border-top: 1px dashed black; padding-top: 5px;"> HDR, SAi1, KEi, Ni --> <-- HDR, SAR1, KEr, Nr, [CERTREQ] </td></tr> </tbody> </table> HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. The SAi1 payload states the cryptographic algorithms the initiator supports for the IKE SA. The KE payload sends the initiator’s Diffie-Hellman value. Ni is the initiator’s nonce. <i>RFC-5996</i> at 10.	Initiator	Responder	HDR, SAi1, KEi, Ni --> <-- HDR, SAR1, KEr, Nr, [CERTREQ]	
Initiator	Responder				
HDR, SAi1, KEi, Ni --> <-- HDR, SAR1, KEr, Nr, [CERTREQ]					

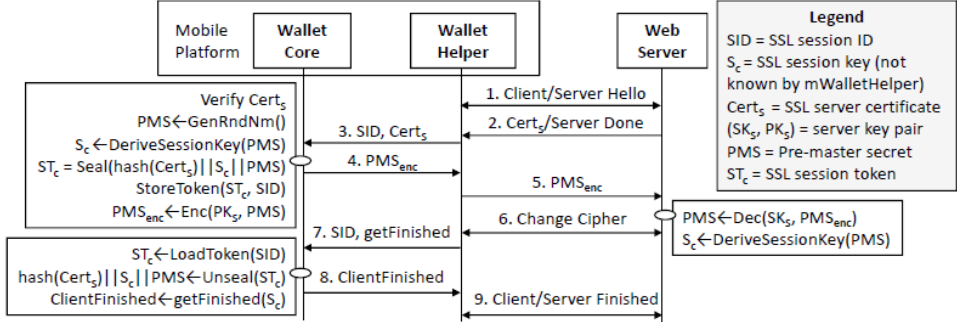
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 1 (“This document describes version 2 of the Internet Key Exchange (IKE) protocol. IKE is a component of IPsec used for performing mutual authentication and establishing and maintaining security associations (SAs).”), 7 (“Communication using IKE always begins with IKE_SA_INIT and IKE_AUTH exchanges (known in IKEv1 as Phase 1). These initial exchanges normally consist of four messages All communications using IKE consist of request/response pairs. . . . The first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange [DH].”); <i>see also RFC-4306</i> at 3-12, 14, 19, 21, 23-24, 26-34, 42-44, 47, 50-52, 55-56, 64, 86, 88-89, 96-97. In addition, at the time of the '838 patent, it was common knowledge to a POSITA to perform a cryptographic key exchange with a remote server to set up a secure communication session using IKE or SSL/TLS protocols. For example, <i>Buer754</i> describes key exchange algorithms in IPsec and SSL, e.g., IKE, RSA public key exchange in SSL. <i>See Buer754</i> at 1:27-41 (IPsec and SSL), 1:42-55 (“The standards and algorithms set forth above require some form of key exchange”), 6:4-31 (describing SSL secure connection may be established using standard public/private key exchange scheme), 8:63-9:4 (Internet Key Exchange protocol), 1:61-2:8 (public/private key exchange session for SSL), 10:14-22 (“Fig. 10 depicts an embodiment that supports asymmetric key exchange”), 10:42-61 (describing asymmetric key exchange), 13:1-15:11 (describing cryptographic key exchange algorithms based on RSA and Diffie-Hellman for establishing secure channel), 15:17-16:46 (describing support for IPsec, SSL and cryptographic algorithms used, such as RSA, Diffie-Hellman, elliptic curve key exchange algorithms); <i>see also Buer754</i> at 1:13-1:27, 2:9-13, 2:17-60, 3:1-11, 3:25-27, 3:49-4:31, 4:39-44, 4:61-5:6, 5:24-62, 6:36-7:9, 7:59-64, 8:63-9:4, 9:16-21, 10:11-22, 11:4-31, 11:40-45, 11:62-12:3, 12:13-23, 17:48-62, Abstract, Figs. 1-4, 10, 12-13, claim 18; <i>Tugenberg</i> at 5:48-61 (“[t]he host processor uses [a] certificate to establish a secure session with, for example, an internet vendor. Secure Socket Layer (SSL), for example, is a standard technique used in the internet community to establish such a session.”), 4:54-58 (“[T]he present invention provides for a method of purchasing items over a non-secure communication link, such as the internet, for example using a secure communication device such as communication device 10 of FIG. 1.”), Abstract, 5:55-61, 1:7-10, 1:26-36, 1:48-55, 2:31-58, 4:20-31, 4:38-49, 4:54-58, 5:10-15, 6:12-20, 6:26-39, 8:7-20; <i>Tkacik</i> ¶¶ [0079]-[0080] (“keys may be generated using any of a number of protocols known to those skilled in the art, including IKE, DH, RSA key gen., SSL, random key, and the like”), Abstract, ¶¶ [0002]-[0003], ¶¶ [0005]-[0008], ¶

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	[0015], ¶ [0021], ¶ [0026], ¶¶ [0028]-[0031], ¶¶ [0037]-[0038], ¶¶ [0048]-[0050], ¶¶ [0052]-[0053], ¶¶ [0055]-[0056], ¶ [0059], ¶¶ [0064]-[0065], ¶ [0078], ¶¶ [0087]-[0089], Figs. 5-8; <i>Paaske</i> ¶ [0055] (“a public key core 350 for performing asymmetric cryptographic operations such as DSA, RSA and Diffie-Hellman”); <i>see also Paaske</i> ¶ [0003], ¶¶ [0006]-[0007], ¶ [0023], ¶¶ [0028]-[0029], ¶ [0031], ¶ [0034], ¶ [0041], ¶ [0051], ¶¶ [0054]-[0056], ¶ [0061], ¶ [0066], ¶ [0078], ¶ [0101], ¶¶ [0112]-[0114], ¶ [0124], ¶ [0141], ¶ [0149], ¶ [0156], ¶ [0168], ¶ [0179] ¶¶ [0217]-[0227], Figs. 1-3, claims 25-26, 30, 33-38; <i>Aaraj</i> at 8:3-4 (ECC), 8:5-6 (RSA, key generator), 8:9 (RSA, ECC, SSL), 8:11 (SSL, cryptographic algorithms such as RSA, ECC), 8:15-16 (SSL, RSA), 8:1-8:2, 8:6-8:9, 8:30-8:31; <i>Gibod</i> ¶¶ [0033]-[0037] (RSA), ¶ [0071] (RSA, DH, TLS), ¶¶ [0069]-[0070]; <i>see also Gibod</i> ¶ [0004], ¶¶ [0007]-[0008], ¶ [0016], ¶ [0019], ¶ [0021], ¶¶ [0032]-[0037], ¶¶ [0040]-[0044], ¶¶ [0046]-[0047], ¶¶ [0053]-[0054], ¶ [0057], ¶¶ [0069]-[0070], claims 5, 8-9, 11, 19. In addition, <i>Weise</i> discloses its hardware supports SSL and IPsec protocols which involve performing a cryptographic key exchange with the remote server. <i>See Weise</i> at ¶ [0010], ¶ [0023], ¶ [0033], ¶ [0035]. <i>Weise</i> further describes performing cryptographic key exchange by disclosing “many cryptographic protocols only use public key encryption as a mechanism to allow participants setting up a secure communication session to exchange secret/private keys.” <i>See Weise</i> at ¶ [0005], ¶ [0034]; <i>see also Weise</i> at Abstract, ¶ [0002], ¶¶ [0004]-[0006], ¶¶ [0008]-[0014], ¶¶ [0016]-[0043], claims 1-12, 14-17, 19; <i>see also Mujtaba</i> at ¶ [0006] (generating a session key), ¶¶ [0052]-[0053], ¶¶ [0055]-[0057], ¶¶ [0067]-[0068], claims 10, 18, 23, Figs. 7-8.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	Exemplary disclosures of the prior art include the following: <i>Bugiel</i> discloses the following at Fig. 2:  Fig. 2. SSL/TLS handshake <i>See also Bugiel</i> at 220-236. <i>See also Chou</i> ¶¶ [0005]-[0006], ¶ [0017], ¶ [0020], ¶¶ [0034]-[0035], ¶¶ [0043]-[0062], ¶ [0066], ¶¶ [0068]-[0070], Figs. 1, 3-5. <i>See also TLS-1.2</i> at 4-6, 23, 29, 33-65, 70-76, 78-83, 86-87, 91-94, 96-100, Figs. 1-2. <i>See also SSL-3.0</i> at 4, 10, 16-35, 37-42, 45-49, 51-52, 55-62. <i>See also TLS-1.0</i> at 4, 20, 22, 25, 27-48, 51-54, 59-62, 64, 66-74, 76, Figs. 1-2. <i>See also TLS-1.1</i> at 4-5, 22, 27, 30-53, 57-60, 66-68, 72, 74-77, 80-83, Figs. 1-2.

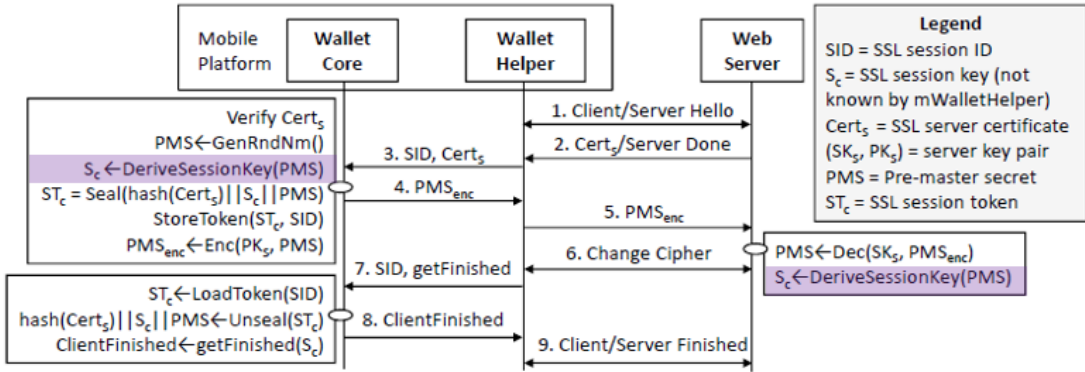
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also RFC-5996</i> at 1, 3, 5-6, 9-23, 25-26, 31-33, 39-41, 44-48, 50-54, 72, 74, 80-83, 86-87, 119-121, 132-137. <i>See also RFC-4306</i> at 1, 3-12, 14, 19, 21, 23-24, 26-34, 42-44, 47, 50-52, 55-56, 64, 86, 88-89, 96-97. <i>See also Buer754</i> at 1:13-1:55, 1:61-2:13, 2:17-60, 3:1-11, 3:25-27, 3:42-4:31, 4:39-44, 4:61-5:6, 5:24-62, 6:4-31, 6:36-7:9, 7:59-64, 8:63-9:4, 9:16-21, 10:11-22, 10:42-61, 11:4-31, 11:40-45, 11:62-12:3, 12:13-23, 13:1-15:11, 15:17-16:46, 17:48-62, Abstract, Figs. 1-4, 10, 12-13, claim 18. <i>See also Tugenberg</i> at Abstract, 1:7-10, 1:26-36, 1:48-55, 2:31-58, 4:20-31, 4:38-49, 4:54-58, 5:10-15, 5:48-61, 6:12-20, 6:26-39, 8:7-20. <i>See also Tkacik</i> Abstract, ¶¶ [0002]-[0003], ¶¶ [0005]-[0008], ¶ [0015], ¶ [0021], ¶ [0026], ¶¶ [0028]-[0031], ¶¶ [0037]-[0038], ¶¶ [0048]-[0050], ¶¶ [0052]-[0053], ¶¶ [0055]-[0056], ¶ [0059], ¶¶ [0064]-[0065], ¶¶ [0078]-[0080], ¶¶ [0087]-[0089], Figs. 5-8. <i>See also Paaske</i> ¶ [0003], ¶¶ [0006]-[0007], ¶ [0023], ¶¶ [0028]-[0029], ¶ [0031], ¶ [0034], ¶ [0041], ¶ [0051], ¶¶ [0054]-[0056], ¶ [0061], ¶ [0066], ¶ [0078], ¶ [0101], ¶¶ [0112]-[0114], ¶ [0124], ¶ [0141], ¶ [0149], ¶ [0156], ¶ [0168], ¶ [0179] ¶¶ [0217]-[0227], Figs. 1-3, claims 25-26, 30, 33-38. <i>See also Aaraj</i> at 8:1-9, 8:11, 8:15-16, 8:30-31. <i>See also Gibod</i> ¶ [0004], ¶¶ [0007]-[0008], ¶ [0016], ¶ [0019], ¶ [0021], ¶¶ [0032]-[0037], ¶¶ [0040]-[0044], ¶¶ [0046]-[0047], ¶¶ [0053]-[0054], ¶ [0057], ¶¶ [0069]-[0071], claims 5, 8-9, 11, 19. <i>See also Weise</i> at Abstract, ¶ [0002], ¶¶ [0004]-[0006], ¶¶ [0008]-[0014], ¶¶ [0016]-[0043], claims 1-12, 14-17, 19. <i>See also Mujtaba</i> at ¶ [0006], ¶¶ [0052]-[0053], ¶¶ [0055]-[0057], ¶¶ [0067]-[0068], claims 10, 18, 23, Figs. 7-8. <i>See supra</i> [1b].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
[1d] generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session;	The grounds alleged at the beginning of this chart disclose or suggest the security engine to “generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session.” The grounds alleged here uses one of <i>Chou</i>, <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i>’s SoC apparatus to implement <i>Bugiel</i>’s communications method, including to implement IKE or SSL/TLS Handshake Protocol messages, to establish a secure communication as explained in the beginning of this chart. <i>Bugiel</i> teaches to (1) generate symmetric session key Sc based on cryptographic key exchange with a remote server according to the TLS/SSL handshake protocol; and (2) use the session key Sc to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session. <i>Chou</i> teaches HSM module 20 (part of the claimed “security engine”), which includes PKA 56 to generate such keys, including those used in TLS/SSL handshake exchanges, and SPU-M module 39 (also part of the claimed “security engine”) to encrypt/decrypt messages using symmetric encryption algorithms, including those used in TLS/SSL exchanges. <i>Bugiel</i> discloses generating a “SSL/TLS session key Sc” (the claimed “symmetric session key”) after “the client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters.” <i>Bugiel</i> at 225. <i>Bugiel</i> also states: “All following messages should be encrypted using Sc.” <i>Id.</i> This includes “messages sent to the remote server” and “messages received from the remote server,” as claimed. Therefore, <i>Bugiel</i> teaches using session key Sc “to encrypt messages sent to the remote server,” as claimed. <i>Bugiel</i>’s Figure 2 also indicates that a remote web server derives the same session key Sc as the client device. <i>See id.</i> at 225-226 (“Sc←DeriveSessionKey (PMS)”). That is, the remote sever uses the same session key Sc to encrypt messages sent to the client device, and the client device uses the same session key Sc to decrypt message received from the remote server during the secure communication session. <i>Bugiel</i> thus teaches the symmetric session key Sc is used “to encrypt messages sent to the

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	remote server and decrypt messages received from the remote server during the secure communication session,” as claimed. <i>See also Bugiel</i> at 225-231, 233, 235-236.  Fig. 2. SSL/TLS handshake <i>Bugiel</i> at Fig. 2 (annotated). Additionally, <i>Chou</i> discloses a PKA module 56 to perform cryptographic key exchange algorithms, such as “Diffie-Hellman public value and shared key computation” or “RSA key generation,” to generate keys. <i>Chou</i> ¶¶ [0043]-[0053]. Figure 3 of <i>Chou</i> illustrates that the PKA module 56 of HSM 20 is communicatively connected to key cache 65 of SPU-M module 39 via bus 71. <i>See Chou</i> ¶ [0035] (“Other units 54-69 are also coupled to bus 71”); ¶ [0054] (“HSM software may access the entire key cache memory from bus 71.”) This indicates how keys generated by PKA module 56 are loaded into key cache 65 for encryption/decryption operations by SPU-M 39. <i>See Chou</i> ¶ [0055] (“Keys for these algorithms may be loaded by HSM 20 into key cache 65.”). A POSITA would have found it obvious to implement <i>Bugiel</i>’s TruWalletM method by using these hardware features in <i>Chou</i>.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	FIG. 3 <i>Chou</i>, Fig. 3 <i>Chou</i> further explains that SPU-M module 39 is used to encrypt/decrypt payload of transmitted messages using key from key cache 65. <i>See Chou</i> ¶ [0055] (“SPU-M module 39 receives the payload and accesses the correct key from key cache 65 and perform decryption/encryption operations”); <i>see also id.</i> ¶ [0021] (“Security processor 21 provides the various processing within HSM 20, including key authentication and the encrypting and/or decrypting operation to encrypt and/or decrypt data. Secure key module 23 is a storage medium, such as a memory, register, circuit, etc. that stores one or more key value(s) that is used by security processor 21 to encrypt and decrypt data.”).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>Chou</i> also explains that SPU-M 39 “operates as a hardware accelerator for symmetric key algorithm.” <i>Chou</i> ¶ [0054]; <i>see also id.</i> ¶ [0055] (“SPU-M module 39 may implement many symmetric key algorithms such as AES, DES, RC4, SHA1-HMAC, SHA2-HMAC, MD5-HMAC, as well as others.”); <i>see also Chou</i> ¶¶ [0005]-[0008], ¶¶ [0017]-[0019], ¶ [0021], ¶¶ [0023]-[0028], ¶ [0035], ¶ [0037], ¶¶ [0043]-[0061], ¶¶ [0069]-[0071], Figs. 1 and 3, claims 1, 4, 6-8, 14, 16, 18-19. This indicates that keys generated by PKA module 56 based on the Diffie-Hellman or RSA key exchange algorithms are symmetric keys. Additionally, each TLS/SSL protocol teaches to generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session in at least two ways. First, an SSL/TLS master secret may be the claimed “symmetric session key” under Daedalus’s apparent view. <i>See</i> Infringement Contentions, Ex. F-1 at 15-19; Ex. F-2 at 17-22. Under such a mapping, <i>TLS-1.2</i> discloses this claim feature by teaching to generate a master secret (the claimed “symmetric session key”) based on the cryptographic key exchange in the TLS handshake process (the claimed “cryptographic key exchange”). <i>See TLS-1.2</i> at 64 (Computing the Master Secret). A POSITA thus would have understood <i>Bugiel</i>’s session key <i>Sc</i> (the claimed “symmetric session key”) may be a TLS master key. <i>See TLS-1.2</i> at 64 (Computing the Master Secret); <i>see also TLS-1.2</i> at 4-6, 16-19, 25-27, 33-37, 58, 63-65, 77-82, 91, 94. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provide similar disclosures. <i>See SSL-3.0</i> at 32 (computing master secret from pre_master_secret); <i>see also SSL-3.0</i> at 11, 29-34, 40, 45, 54-59; <i>TLS-1.0</i> at 47 (computing the master secret); <i>see also TLS-1.0</i> at 14-15, 21-23, 29, 45-48, 56, 59, 68-72; <i>TLS-1.1</i> at 52 (computing the master secret); <i>see also TLS-1.1</i> at 15-16, 24-25, 31, 48, 51-53, 64, 66, 74-78. Second, and alternatively, <i>TLS-1.2</i> also discloses generating encryption keys (e.g., a key_block) for data encryption, based on the master secret for each session. <i>TLS-1.2</i> at 25-26 (Key Calculation).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<pre>key_block = PRF(SecurityParameters.master_secret, "key expansion", SecurityParameters.server_random + SecurityParameters.client_random);</pre> The key_block is a key with multiple partitions, including a “client_write_key” partition and a “server_write_key” partition. <i>See id.</i> at 26. The client uses the client_write_key partition to encrypt data transmitted from client to server, and the client uses the server_write_key partition to decrypt data received from the server. <i>See id.</i> at 79-81 (defining the master_secret, client_write_key partition, and server_write_key partition). <i>TLS-1.2</i> further explains, “TLS secures transmitted application data via the use of symmetric encryption,” thereby indicating the encryption keys are symmetric session keys. <i>Id.</i> at 95; <i>see also TLS-1.2</i> at 4 (“Symmetric cryptography is used for data encryption The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by another protocol (such as the TLS Handshake Protocol).”), 6, 12-14, 16-18, 22-28, 33, 35, 40, 58, 63, 65, 69, 75-84, 86, 91, 94-100. A POSITIA thus would have understood <i>Bugiel</i>’s session key Sc (the claimed “symmetric session key”) may be a TLS key_block. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> also provides similar disclosures as <i>TLS-1.2</i>. <i>See SSL-3.0</i> at 33-34 (computing key_block based on master secret), 44-46 (defining the master secret, client_write_key partition, and server_write_key partition), 4 (“Encryption is used after an initial handshake to define a secret key. Symmetric cryptography is used for data encryption (e.g., DES[DES], RC4[RC4], etc.)”); <i>see also SSL-3.0</i> at 9-11, 14-17, 19, 29-37, 42-46, 48, 50-51, 55, 58-59, 61; <i>see TLS-1.0</i> at 21-23 (key calculation), 57-60 (defining the master secret, client_write_key partition, and server_write_key partition), 3 (“Symmetric cryptography is used for data encryption The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by another protocol (such as the TLS Handshake Protocol).”); <i>see also TLS-1.0</i> at 4, 10, 12-16, 18-19, 21-23, 26, 30, 35, 46, 49, 55-60, 62, 64, 69, 72-73, 76; <i>TLS-1.1</i> at 24 (key calculation), 64-68 (defining the master secret, client_write_key partition, and server_write_key partition), 4 (“Symmetric cryptography is used for data encryption The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by another protocol (such as the TLS Handshake Protocol).”); <i>see also TLS-1.1</i> at 6, 11, 13, 15-17, 20, 24-26, 28, 33, 37, 55, 63-70, 75, 78-81.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	The combination based on SSL/TLS protocols alleged here incorporates one of the SSL/TLS protocols (<i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i>, or <i>TLS-1.2</i>)'s teaching (e.g., the <i>TLS-1.2</i> as referenced in <i>Bugiel</i>) to (1) generate an encryption key (either “master secret” or “key_block” as described above) based on cryptographic key exchange with a remote server according to the TLS handshake protocol; and (2) use the same session key (either “master secret” or “key_block”) to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session. Additionally, in the combinations based on IPsec IKE protocols (<i>RFC-5996</i> or <i>RFC-4306</i>), each of the IPsec IKE protocol teaches the claimed “generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session.” For example, <i>RFC-5996</i> discloses generating a shared secret g^{ir} in the initial exchange or generating a g^{ir} (new) in CHILD_CREATE_SA exchange “from the ephemeral Diffie-Hellman exchange.” <i>RFC-5996</i> at 46-47, 52-53, 5; <i>see also RFC-4306</i> at 28-29, 33-34, 3. The initiator then calculates a “quantity called SKEYSEED . . . from the nonces exchanged during the IKE_SA_INIT exchange and the Diffie-Hellman shared secret established during that exchange. SKEYSEED is used to calculate seven other secrets: SK_d used for deriving new keys for the Child SAs established with this IKE SA; SK_{ai} and SK_{ar} used as a key to the integrity protection algorithm for authenticating the component messages of subsequent exchanges; SK_{ei} and SK_{er} used for encrypting (and of course decrypting) all subsequent exchange; and SK_{pi} and SK_{pr}, which are used when generating an AUTH payload.” <i>RFC-5996</i> at 46-47. Keying material KEYMAT for Child SAs is generated either based on SK_d or SK_d and g^{ir} (new). <i>See RFC-5996</i> at 52; <i>see also RFC-5996</i> at 11-13, 46-47, 52-53.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	2.17. Generating Keying Material for Child SAs A single Child SA is created by the IKE_AUTH exchange, and additional Child SAs can optionally be created in CREATE_CHILD_SA exchanges. Keying material for them is generated as follows: KEYMAT = prf+(SK_d, Ni Nr) Where Ni and Nr are the nonces from the IKE_SA_INIT exchange if this request is the first Child SA created or the fresh Ni and Nr from the CREATE_CHILD_SA exchange if this is a subsequent creation. For CREATE_CHILD_SA exchanges including an optional Diffie-Hellman exchange, the keying material is defined as: KEYMAT = prf+(SK_d, g^{ir} (new) Ni Nr) where g^{ir} (new) is the shared secret from the ephemeral Diffie-Hellman exchange of this CREATE_CHILD_SA exchange (represented as an octet string in big endian order padded with zeros in the high-order bits if necessary to make it the length of the modulus). <i>RFC-5996</i> at 52. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 8-10, 28-29, 33-34, 55. Under Daedalus's apparent view, the initiator generates these symmetric keys g^{ir}, g^{ir} (new), SKEYSEED, SK_d, SK_ai, SK_ar, SK_ei, SK_er, KEYMAT, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session. In addition, at the time of the '838 patent, it was well-known to a POSITA to generate a symmetric session key, based on the cryptographic key exchange, to encrypt message sent to the remote server and decrypt messages received from the remote server during the secure communication session. For example, <i>Buer754</i> describes "Initiator derives shared key material" based on the cryptographic key exchange using Diffie-Hellman, RSA, or Shared Secret ReKey algorithms. <i>Buer754</i> at 13:1-15:32, 4:14-26 (security association used to refer to key information, which may include key(s) or encrypted key(s)); <i>see also Buer754</i> at Abstract, 1:13-49, 1:61-2:13, 2:17-40, 2:50-60, 3:58-4:38, 5:7-51, 6:4-35, 6:53-7:64, 8:1-3, 8:63-9:44, 11:4-39, 11:62-12:3, 12:53-67, 17:27-37, 17:48-62, Figs. 1-2, 4-6, 12, claims 1-9, 14, 16; <i>see also Tugenberg</i> at 5:48-61 ("[t]he host processor uses [a] certificate to establish

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	a secure session with, for example, an internet vendor. Secure Socket Layer (SSL), for example, is a standard technique used in the internet community to establish such a session.”), 4:32-42 (“[T]he decrypted secret key may be the user’s private key which is part of a public/private key pair. . . . [T]he secret key may be secret information used to generate a common session key with another communication device. The secret key may also be the common session key used to subsequently communicate secured information.”). Abstract, 5:55-61, 1:7-10, 1:26-36, 1:48-55, 2:31-58, 4:20-31, 4:38-49, 4:54-58, 5:10-15, 6:12-20, 6:26-39, 8:7-20; <i>Tkacik</i> ¶¶ [0079]-[0080] (generating a plaintext active key 172 (the claimed “symmetric session key”) based on generated random number using any of a number of protocols known to those skilled in the art, including IKE and SSL); <i>see also Tkacik</i> Abstract, ¶ [0009], ¶ [0025], ¶ [0052], ¶¶ [0054]-[0056], ¶¶ [0064]-[0068], ¶ [0071], ¶¶ [0073]-[0075], ¶ [0077], ¶¶ [0080]-[0082], ¶¶ [0084]-[0085], ¶¶ [0087]-[0089]; <i>Paaske</i> ¶ [0031] (“TPM may generate and maintain keys” and “may include one or more cryptographic processors that perform cryptographic operations including, for example, one or more of encryption, decryption, authentication, verification, signing and key management.”), ¶ [0055] (“may include a symmetric cryptographic core 354 for performing algorithms such as DES, 3DES and AES.”), ¶ [0124] (explaining that “in an embodiment that supports 3DES and HMAC the system may initialize a 3DES key for decryption operations”); ¶ [0043] (“[T]he information received by the TPM 202 may be encrypted by the trusted source or the computing system 200. In this case, a key securely stored in the computing system (e.g., in the TPM 202, the integrated circuit 204 or encrypted in the flash memory 206) may be used to decrypt the received information.”), ¶ [0029] (explaining the “trusted source may comprise an external (e.g., remote) secure signing environment”), ¶ [0006] (“[W]hen a server receives a message from a remote client, authentication information associated with the message may be used to verify that the message is from a specific source. In this way, the server may ensure that only authorized clients access the applications and data provided by the server.”), ¶¶ [0221]-[0227]; <i>Aaraj</i> at 8:6 (“TPM’s random number generator (RNG), key generator and RSA engine, support the generation of RSA and symmetric keys, and RSA cryptographic operations, such as signing and encryption/decryption.”), 8:2 (“cryptographic key generation”), 8:5 (“cryptographic key generation”), 8:7-8:8 (“key generation”), 8:11 (SSL), 8:12 (“exchange the symmetric key used for bulk (voice) data encryption.”), 8:14 (“symmetric session key is generated, encrypted with both (Z)’s and (C)’s AIKs, and sent to them”), 8:15-8:16 (SSL to generate session key); <i>see also Gibod</i> ¶ [0071] (TLS, symmetric key), ¶¶ [0004]-[0009] (secret information includes cryptographic keys), ¶ [0016], ¶ [0019], ¶ [0021],

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	¶ [0025], ¶¶ [0027]-[0031], ¶¶ [0033]-[0037], ¶ [0040], ¶ [0045], ¶ [0047], ¶¶ [0051]-[0055], ¶ [0058], ¶¶ [0060]-[0064], ¶ [0067], ¶¶ [0069]-[0070], claims 1, 6-7, 15, 19, Figs. 1-5; <i>see also Mujtaba</i> at ¶ [0006] (generating a session key), ¶¶ [0052]-[0053], ¶¶ [0055]-[0057], ¶¶ [0067]-[0068], claims 10, 18, 23, Figs. 7-8; <i>see also Weise</i> at ¶ [0010] (describing its hardware supports SSL protocol which involves generating a symmetric session key, based on the cryptographic key exchange, for encryption and decryption), ¶ [0023] (SSL), ¶ [0033] (SSL), ¶ [0035] (SSL), ¶ [0005]-[0006] (describing its HSM hardware can use the symmetric session key for encrypting data in secure communication session); <i>see also Weise</i> ¶¶ [0008]-[0009], ¶ [0011], ¶¶ [0013]-[0014], ¶¶ [0021]-[0024], ¶ [0034], ¶¶ [0042]-[0043], Abstract, claims 1-2, 9, 12, 14, 17, 19, Figs. 1, 3-5. Exemplary disclosures of the prior art include the following: <i>Bugiel</i> discloses the following at 225-227 (Establishing SSL/TLS connection and managing two logical SSL/TLS subchannels): Figure 2 illustrates the SSL handshake protocol. The client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters. In the following steps the server's certificate Certs is sent and verified, the SSL/TLS session key Sc is generated (which is never disclosed to WalletHelper) and sealed (together with pre-master secret PMS and the hash value hash(Certs)). The resulting session-token STc is stored in a secure storage and can be loaded and unsealed later using the session identifier SID. ... ChangeCipher messages exchanged by WalletHelper and WebServer indicate they are ready to switch to the newly-negotiated parameters and the secret key. All following messages should be encrypted using Sc.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

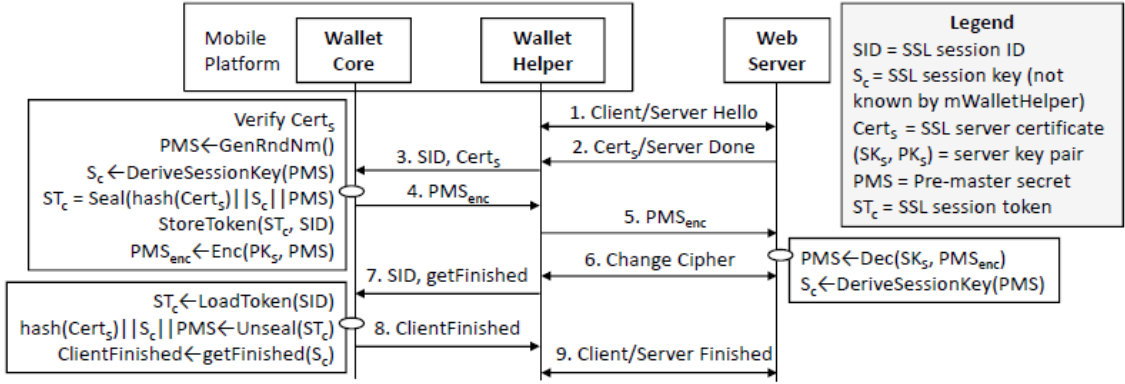
The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16].  The diagram illustrates the SSL/TLS handshake process involving four entities: Mobile Platform, Wallet Core, Wallet Helper, and Web Server. The process follows these steps: 1. Client/Server Hello (Mobile Platform to Web Server) 2. Cert_s/Server Done (Web Server to Wallet Helper) 3. SID, Cert_s (Wallet Helper to Wallet Core) 4. PMS_{enc} (Wallet Core to Wallet Helper) 5. PMS_{enc} (Wallet Helper to Web Server) 6. Change Cipher (Web Server to Wallet Helper) 7. SID, getFinished (Wallet Helper to Wallet Core) 8. ClientFinished (Wallet Core to Wallet Helper) 9. Client/Server Finished (Wallet Helper to Web Server) Legend: - SID = SSL session ID - S_c = SSL session key (not known by mWalletHelper) - Cert_s = SSL server certificate (SK_s, PK_s) = server key pair - PMS = Pre-master secret - ST_c = SSL session token Mobile Platform actions: - Verify Cert_s - PMS ← GenRndNm() - S_c ← DeriveSessionKey(PMS) - ST_c = Seal(hash(Cert_s) S_c PMS) - StoreToken(ST_c, SID) - PMS_{enc} ← Enc(PK_s, PMS) - ST_c ← LoadToken(SID) - hash(Cert_s) S_c PMS ← Unseal(ST_c) - ClientFinished ← getFinished(S_c) Web Server actions: - PMS ← Dec(SK_s, PMS_{enc}) - S_c ← DeriveSessionKey(PMS)

Fig. 2. SSL/TLS handshake

See also *Bugie* at 225-231, 233, 235-236.

See also *Chou* ¶¶ [0005]-[0008], ¶¶ [0017]-[0019], ¶ [0021], ¶¶ [0023]-[0028], ¶ [0035], ¶ [0037], ¶¶ [0043]-[0062], ¶¶ [0069]-[0071], Figs. 1 and 3, claims 1, 4, 6-8, 14, 16, 18-19.

See also *TLS-1.2* at 4-6, 12-14, 16-19, 22-28, 33-37, 40, 58, 63-65, 69, 75-84, 86, 91, 94-100.

See also *SSL-3.0* at 4, 9-11, 14-17, 19, 29-34, 40, 42-46, 48, 50-51, 54-59, 61.

See also *TLS-1.0* at 3-4, 10, 12-16, 18-19, 21-23, 26, 29-30, 35, 45-49, 55-60, 62, 64, 68-73, 76.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also TLS-1.1</i> at 4, 6, 11, 13, 15-17, 20, 24-26, 28, 31, 33, 37, 48, 51-53, 55, 63-70, 74-81. <i>See also RFC-4306</i> at 3, 5, 11-13, 28-29, 33-34, 46-47, 52-53. <i>See also RFC-4306</i> at 3, 8-10, 28-29, 33-34, 55. <i>See also Buer754</i> at Abstract, 1:13-49, 1:61-2:13, 2:17-40, 2:50-60, 3:42-48, 3:58-4:38, 5:7-51, 6:4-35, 6:53-7:64, 8:1-3, 8:63-9:44, 11:4-39, 11:62-12:3, 12:53-67, 13:1-15:32, 17:27-37, 17:48-62, Figs. 1-2, 4-6, 12, claims 1-9, 14, 16. <i>See also Tugenberg</i> at Abstract, 1:7-10, 1:26-36, 1:48-55, 2:31-58, 4:20-49, 4:54-58, 5:10-15, 5:48-61, 6:8-20, 6:26-39, claims 1, 5, 12, Figs. 1 and 3. <i>See also Tkacik</i> Abstract, ¶ [0009], ¶ [0025], ¶ [0052], ¶¶ [0054]-[0056], ¶¶ [0064]-[0068], ¶ [0071], ¶¶ [0073]-[0075], ¶¶ [0077]-[0082], ¶¶ [0084]-[0085], ¶¶ [0087]-[0089], Figs. 1-2, 5-8. <i>See also Paaske</i> ¶ [0006], ¶ [0029], ¶ [0031], ¶ [0043], ¶ [0055], ¶ [0124], ¶¶ [0221]-[0227], Figs. 2-3. <i>See also Aaraj</i> at 8:2, 8:5-8:8, 8:11-16, 8:30-31. <i>See also Gibod</i> ¶¶ [0004]-[0009], ¶ [0016], ¶ [0019], ¶ [0021], ¶ [0025], ¶¶ [0027]-[0031], ¶¶ [0033]-[0037], ¶ [0040], ¶ [0045], ¶ [0047], ¶¶ [0051]-[0055], ¶ [0058], ¶¶ [0060]-[0064], ¶ [0067], ¶¶ [0069]-[0071], claims 1, 6-7, 15, 19, Figs. 1-5. <i>See also Mujtaba</i> at ¶ [0006], ¶¶ [0052]-[0053], ¶¶ [0055]-[0057], ¶¶ [0067]-[0068], claims 10, 18, 23, Figs. 7-8. <i>See also</i> disclosures of [1b], [1c].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
[1e] encrypt the symmetric session key based on the security key; and	The grounds alleged at the beginning of this chart disclose or suggest the security engine to “encrypt the symmetric session key based on the security key.” <i>Bugiel</i> teaches to encrypt the symmetric session key S_c (the claimed “symmetric session key”) using another key (the claimed “security key”) that is never available outside the TrEE. <i>See Bugiel</i> at 225-226 & n. 8 (explaining that “[s]ealing means protecting an object so that only a certain set of entities can access or use it”), 231 (explaining that “[w]hen sealed, data is encrypted under a key that is never available outside the TrEE”); <i>see also id.</i> at 226-230, 232-233, 235-236, Fig. 2. Therefore, <i>Bugiel</i> teaches this claim feature. <i>See also supra</i> [1a-3]. <pre> sequenceDiagram participant MP as Mobile Platform participant WC as Wallet Core participant WH as Wallet Helper participant WS as Web Server Note over MP: Verify Cert_s PMS ← GenRndNm() Note over MP: S_c ← DeriveSessionKey(PMS) Note over MP: ST_c = Seal(hash(Cert_s) S_c PMS) Note over MP: StoreToken(ST_c, SID) Note over MP: PMS_enc ← Enc(PK_s, PMS) MP->>WH: 3. SID, Cert_s WH->>WS: 1. Client/Server Hello WS->>WH: 2. Cert_s/Server Done WH->>MP: 4. PMS_enc MP->>WS: 5. PMS_enc WS->>WH: 6. Change Cipher WH->>MP: 7. SID, getFinished Note over WS: PMS ← Dec(SK_s, PMS_enc) Note over WS: S_c ← DeriveSessionKey(PMS) MP->>WH: 8. ClientFinished Note over MP: ST_c ← LoadToken(SID) Note over MP: hash(Cert_s) S_c PMS ← Unseal(ST_c) Note over MP: ClientFinished ← getFinished(S_c) WH->>WS: 9. Client/Server Finished </pre> Legend SID = SSL session ID S_c = SSL session key (not known by mWalletHelper) Cert_s = SSL server certificate (SK_s, PK_s) = server key pair PMS = Pre-master secret ST_c = SSL session token Fig. 2. SSL/TLS handshake In addition, as explained for claim feature [1a-3], a POSITA would have found it obvious to use <i>Chou</i>’s device unique key/root key as this claimed “security key.” A POSITA would have found it obvious to use <i>Chou</i>’s HSM 20 (part of the claimed “security engine”) to implement this aspect of <i>Bugiel</i>’s method, where KEK 55 would encrypt the symmetric session key S_c using <i>Chou</i>’s device unique key/root key stored in OTP 54. Accordingly, in the combination of <i>Bugiel</i>, <i>Chou</i>, and optionally one of the know protocols, e.g., SSL/TLS or IPsec IKE, etc., <i>Chou</i>’s HSM 20 (part of the claimed “security engine”) encrypts the session key S_c (the claimed “symmetric session key”) based on <i>Chou</i>’s device unique key/root key (the claimed “security key.”). <i>See Chou</i> Abstract, ¶ [0003],

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	¶ [0005], ¶¶ [0007]-[0009], ¶¶ [0017]-[0019], ¶ [0021], ¶¶ [0023]-[0028], ¶¶ [0034]-[0041], ¶¶ [0054]-[0055], ¶ [0063], ¶¶ [0069]-[0071], claims 1, 4, 6, 8, 19, Figs. 1 and 3. In addition to <i>Chou</i>, a POSITA would have also found it obvious to use the security key disclosed in one of <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i> to encrypt the session key in <i>Bugiel</i>. <i>See supra</i> [1a-3]. For example, <i>Tkacik</i> teaches using the master key 160 (the claimed “security key”) to encrypt the plaintext key 172 (the claimed “symmetric session key”) to form a long-lived ciphertext key (the claimed “encrypted session key”). <i>Tkacik</i> ¶¶ [0083]-[0084], ¶ [0046]. As explained for claim feature [1a-3], <i>Tkacik</i> teaches a plain master key 160 that is a “security key,” as claimed, by explaining that “long-lived key register 60 is desirably programmed with plaintext master key 160 during the manufacturing process of secure processing system 12 and cannot thereafter be altered.” <i>Id.</i> ¶ [0046]. <i>Tkacik</i> also explains that plain master key 160 can “directly serve[] as the BKEK” if desired. <i>Id.</i> ¶ [0083]. Here, BKEK means “blob key encryption key.” <i>Id.</i> ¶ [0069]. The BKEK is a cryptographic key to be used in forming a blob, “a self-contained object which includes a cryptographic key in an encrypted form and a message authentication code (MAC).” <i>Id.</i> ¶ [0070]. <i>Tkacik</i> further teaches generating the plaintext active key 172 (the claimed “symmetric session key”) using IKE or SSL. <i>See supra</i> [1d]; <i>Tkacik</i> ¶ [0080]. Referring to process 140 in <i>Tkacik</i>’s Figure 8, <i>Tkacik</i> describes moving the generated plaintext active key 172 (the claimed “symmetric session key”) “to data register 76 and encrypt[ing] this key 172 using . . . the BKEK” (the plaintext master key 160, i.e., the claimed “security key”). <i>Tkacik</i> ¶ [0084]. According to <i>Tkacik</i>, the secure process section 30 performs this encryption. <i>See Tkacik</i> ¶ [0079] (“Process 140 may be performed by controller 46 in response to the execution of software programming instructions using components of secure processing section 30, and possibly other components of host device 10.”).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<pre> graph TD 140([MAKE NEW KEY]) --> 142[GENERATE RANDOM NUMBER] 142 --> 144[COMBINE WITH MODIFIER COMPONENTS] 144 --> 146[DERIVE SKEK FROM RESET EPOCH KEY AND MODIFIER COMPONENTS] 146 --> 148[ENCRYPT PLAINTEXT KEY AS DATA USING WF ALGORITHM AND SKEK] 148 --> 150[SAVE CIPHERTEXT KEY IN KEY MAGAZINE] 150 --> 152{CONVERT TO LONG-LIVED KEY} 152 -- Y --> 158[DERIVE BKEK FROM MASTER KEY AND MODIFIER COMPONENTS] 158 --> 162[ENCRYPT PLAINTEXT KEY AS DATA USING SS ALGORITHM AND BKEK] 162 --> 164[SAVE CIPHERTEXT KEY IN UNSECURED MEMORY] 152 -- N --> 156[RETAIN ORIGINAL PLAINTEXT KEY IN ACTIVE KEY REGISTER] 164 --> 156 156 --> EXIT([EXIT]) </pre> FIG. 8 <i>Tkacik</i>, Fig. 8 Therefore, <i>Tkacik</i> discloses or suggests the secure processing section 30 (the claimed “security engine”) to encrypt the plaintext active key 172 (the claimed “symmetric session key”) based on the BKEK, i.e., the plaintext master key 160 (the claimed “security key.”). <i>See also Tkacik</i> Abstract, ¶¶ [0002]-[0004], ¶¶ [0006]-[0009], ¶ [0019], ¶ [0029], ¶ [0039], ¶¶ [0044]-[0046], ¶¶ [0048]-[0054], ¶ [0057], ¶¶ [0069]-[0080], ¶¶ [0082]-[0085], ¶¶ [0088]-[0089], Figs. 2, 5-8. In its infringement contentions, Daedalus argues encryption performed using keys <i>derived from</i> the security key satisfy this claim feature. <i>See</i> Infringement Contentions, Ex. F-1 at 19-21. Although Daedalus is wrong, <i>Paaske</i> nevertheless teaches session key encryption using either (1) the security key itself (i.e., as the '838 patent requires) or (2) keys derived from the security key (Daedalus’s infringement theory). <i>See Paaske</i> ¶ [0056] (“Through the use of these keys and/or keys derived from these keys, the TPM may securely store information outside of the TPM (or the integrated circuit

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	within which the TPM resides). For example, the cryptographic processor(s) 330 may use these keys to encrypt and/or authenticate any data that is stored in an external memory such as the flash memory 306.”), ¶ [0110] (“[T]he keys stored in the OTP memory may be used to encrypt other keys so that these other keys may be securely stored in external memory.”), ¶ [0112] (“[T]he TPM (e.g., cryptographic processor(s) 330 in FIG. 3) generates encryption and authentication keys that may be used to encrypt and authenticate information stored in the flash memory. These keys may be securely stored in the flash memory (e.g., in the TPM keys/data section 356) by encrypting the keys using the internal keys.”), ¶ [0114] (“The TPM (e.g., cryptographic processor(s) 330) uses the encryption and authentication keys to encrypt and/or authenticate the flash secure code information. This information may include, for example, . . . the public verification and encryption keys. . . .”), ¶ [0124] (“[T]he system may initialize a 3DES key for decryption operations . . . these keys may be subsequently encrypted using an internal key and stored in the flash memory (e.g., TPM keys/data 356 in FIG. 3).”); <i>see also Paaske</i> ¶ [0029], ¶ [0031], ¶¶ [0043]-[0044], ¶ [0056], ¶¶ [0059]-[0060], ¶ [0101], ¶ [0103], ¶¶ [0111]-[0112], ¶ [0141], ¶ [0149], ¶ [0188], ¶ [0190], ¶¶ [0217]-[0227], Figs. 2-3. In addition, <i>Tugenberg</i>’s Figure 1 shows that secure memory 20 (the claimed “secure memory”) includes a laser-scribed encryption key 21 (the claimed “security key”). <i>See Tugenberg</i> at 2:61-63 (“Secure memory 20 also includes laser-scribed encryption key 21 coupled to blocking gates 22.”), 6:16-18 (describing “a secure memory that includes the laser-scribed encryption key”), claim 1 (reciting “a secure memory that includes a laser-scribed encryption key”), Figs. 1-3. <i>Tugenberg</i> teaches using laser-scribed encryption key 21 (the claimed “security key”) to encrypt a secret key / common session key (the claimed “symmetric session key”). <i>See Tugenberg</i> at 4:23-27 (“[E]ach secret key [is] encrypted with laser-scribed encryption key 21 using the secure processing system 14. When needed, communication device decrypts one of the secret keys with encryption logic circuitry 23 using laser-scribed encryption key 21”), 2:12-15 (“The secure memory uses a laser-scribed encryption key coupled to encryption logic circuitry within the secure memory for encrypting and decrypting sensitive information.”), 2:64-66 (“Encryption logic circuitry 23 preferably implements a symmetric encryption algorithm using laser-scribed encryption key 21.”); <i>see also Tugenberg</i> at

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Abstract, 1:40-55, 2:3-30, 2:59-3:49, 3:58-4:9, 4:20-31, 4:39-58, 5:1-9, 5:47-6:2, 6:8-7:67, claims 1-2, 5-8, 10-12, 14-19, Figs. 1-3. In addition, <i>Aaraj</i> discloses “the endorsement key (EK), is generated by the vendor or manufacturer, and stored in the nonvolatile memory within the TPM. EK should never be disclosed and is only used for . . . creating the storage root key (SRK). . . . The SRK is the root of the hierarchy of TPM keys, and is used for encrypting other migratable keys that are stored outside the TPM.” <i>Aaraj</i> at 8:6; <i>see also Aaraj</i> at 8:5, 8:7-9, 8:13-16, 8:30-31. Therefore, <i>Aaraj</i> teaches this claim feature under Daedalus’s apparent view. In addition, <i>Gibod</i> teaches this claim feature by disclosing a device key/Master key that is “built-in” permanently in the device. <i>Gibod</i> ¶ [0051]; <i>see also Gibod</i> ¶ [0021], ¶ [0023], ¶ [0037], Fig. 1. <i>Gibod</i> also discloses the Master key is stored in secure memory, e.g., “secure storage.” <i>Gibod</i> ¶ [0026], claim 14. <i>Gibod</i> further discloses to use the Master key to encrypt/decrypt other keys. <i>See Gibod</i> ¶¶ [0009]-[0010] (“[t]o export or persist secret information, the secret information may first be encrypted using a persistent (e.g., permanent) encryption key.”), ¶ [0019] (“ . . . secret information (e.g., cryptographic keys) . . .”), ¶ [0004] (“ . . . information (e.g., cryptographic keys, . . .) . . . designed to prevent unauthorized access and/or use to secure and/or proprietary content may also need to be kept secret”), ¶ [0008] (“performing encryption/decryption operations utilizing the secret information, . . . , encrypting secret fields within a larger piece of data utilizing the secret information”), ¶ [0032], ¶ [0004], ¶ [0016], ¶¶ [0021]-[0024], ¶¶ [0026]-[0032], ¶ [0046], ¶¶ [0050]-[0051], ¶ [0053], ¶¶ [0069]-[0070], claims 11-14. In addition, <i>Mujtaba</i> teaches this claim feature by disclosing “Storage key 114 and/or the UID (not shown) may be hardwired in the hardware (e.g., burnt fuses) during the manufacturing of wireless processor 101.”). <i>Mujtaba</i> ¶ [0031], ¶ [0033], ¶ [0037], ¶¶ [0066]-[0068], Figs. 1 and 4. Figures 1 and 4 illustrate the storage key 114 is inside secure memory accessible only by the security engine wireless processor 101. <i>Mujtaba</i> further discloses the storage key or another key derived from the UID is used to encrypt a generated session key. <i>See Mujtaba</i> ¶ [0006], ¶ [0031], ¶¶ [0042]-[0044], ¶¶ [0052]-[0053], ¶ [0055], ¶ [0057], ¶¶ [0066]-[0068], claims 10, 18, 23.

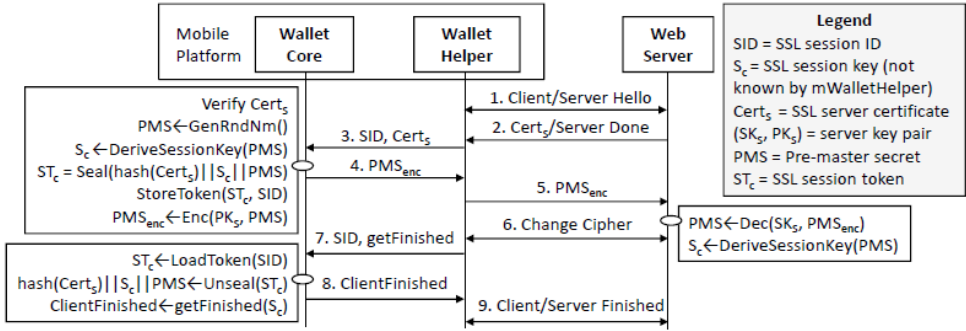
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	In addition, <i>Buer754</i> discloses “these session keys are encrypted using KEK 238 and stored in a database 240 as security associations.” <i>Buer754</i>, 5:43-51; 6:36-43 (“To protect these keys, the security module 320 encrypts them with a KEK 332 before they are sent to the cryptographic accelerator 326”), 2:50-60 (“using a single KEK to encrypt all the keys associated with the network data”), Fig. 4 (encrypted packet & security association sent to cryptographic accelerator 420), Fig. 6 (step 610: encrypt and store session keys), Fig. 11 (steps 1118-1120 : encrypt and store private keys using KEK), Fig. 12 (steps 1210-1212: encrypt session keys using KEK_A and store security associations), 4:45-60 (“the initial KEK 130 may be programmed into the non-volatile memory 134 when the system is manufactured.”); <i>see also Buer754</i> at Abstract, 1:42-49, 1:61-2:3, 2:9-12, 2:25-60, 3:54-56, 4:14-60, 5:43-6:12, 6:36-43, 6:53-64, 6:65-7:23, 7:35-41, 8:1-8, 8:35-62, 9:5-12, 9:22-44, 10:23-27, 10:34-37, 10:62-11:3, 11:17-31, 11:62-12:3, 12:47-52, 13:66-14:10, 15:17-33, 16:8-15, 17:1-26, 17:43-62; Figs. 3, 5-6, 8, 11-13, claims 1-3, 6-9, 12, 14, 16. In addition, at the time of the '838 patent, it was well-known to a POSITA to protect the secure keys such as session key, pre-master key by encrypting them. For example, <i>Weise</i> discloses “encrypt the at least one cryptographic key using a master key.” <i>Weise</i>, claim 12; Abstract, ¶ [0012], ¶ [0037], claim 19. <i>Weise</i> explains that “cryptographic keys . . . are in the context of symmetrical keys as used in accordance with the Data Encryption Standard (DES) as well as other cryptography standard known to those skilled in the art” and that cryptographic keys “may be used herein to refer to any information which it is necessary to be in possession of or use in order that a secure operation can be performed in conjunction with corresponding complementary data providing a useful result.” <i>Weise</i>, ¶ [0006]. <i>Weise</i> also describes that its composite HSM supports SSL and IPsec protocols. <i>See Weise</i>, ¶ [0010], ¶ [0023], ¶ [0033], ¶ [0035]. Therefore, a POSITA would have understood the cryptographic keys in <i>Weise</i> could include the symmetric session key or other keys generated in SSL or IPsec protocols (e.g., the pre-master key). <i>See also Weise</i> Abstract, ¶ [0002], ¶¶ [0005]-[00014], ¶¶ [0021]-[0025], ¶ [0027], ¶ [0030], ¶ [0032], ¶¶ [0034]-[0043], claims 1, 12-13, 19. Thus, <i>Weise</i> teaches encrypting symmetric session key using a master key (the claimed “security key”).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	As explained for feature [1d], in the combinations based on SSL/TLS protocol (<i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i> or <i>TLS-1.2</i>), the symmetric session key <i>Sc</i> may be either (1) a “master secret” or (2) a “key_block,” which is used to encrypt/decrypt messages between client and server. <i>See supra</i> [1c]-[1d]. In the combinations based on IKE protocol (<i>RFC-5996</i> or <i>RFC-4306</i>), the symmetric session key could be any of the generated keys based on the initial or Child SA exchanges, such as g^{ir}, g^{ir} (new), <i>SKEYSEED</i>, <i>SK_d</i>, <i>SK_ai</i>, <i>SK_ar</i>, <i>SK_ei</i>, <i>SK_er</i>, <i>KEYMAT</i>, etc. under Daedalus’s apparent view. The SSL/TLS protocols also express the needs to protect the keys used in the secure connection, including the session keys, by keeping the keys secure. <i>See TLS-1.2</i> at 96 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”) (emphases added); <i>see also TLS-1.2</i> at 1, 4, 16, 34, 91-97. <i>SSL-3.0</i>, <i>TLS-1.0</i> and <i>TLS-1.1</i> also provide similar disclosures. <i>See SSL-3.0</i> at 59 (“For SSL to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 29 (“The client generates session keys, wrap them using the TEK [, which is a Token Encryption Key derived under FORTEZZA key exchange], and sends the results to the server”), 1, 9, 55-58, 17; <i>TLS-1.0</i> at 73 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 13, 29-30, 69, 71-72; <i>TLS-1.1</i> at 80 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 14-15, 31, 74-75, 77-80. In addition, the IKE protocol for IPsec also desires protecting the keys. <i>See RFC-5996</i> at 120-123 (Security Considerations); <i>see also RFC-5996</i> at 45, 51. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 88-90 (Security Considerations), 32.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Therefore, the grounds alleged here teaches encrypting the symmetric session keys based on the security key. Exemplary disclosures include the following: <i>Bugiel</i> discloses the following at Fig. 2:  Fig. 2. SSL/TLS handshake <i>See also Bugiel</i> at 225-233, 235-236. <i>See also Chou</i> Abstract, ¶ [0003], ¶ [0005], ¶¶ [0007]-[0009], ¶¶ [0017]-[0019], ¶ [0021], ¶¶ [0023]-[0028], ¶¶ [0034]-[0041], ¶¶ [0054]-[0063], ¶¶ [0069]-[0071], claims 1, 4, 6, 8, 19, Figs. 1 and 3. <i>See also Tkacik</i> Abstract, ¶¶ [0002]-[0004], ¶¶ [0006]-[0009], ¶ [0019], ¶ [0029], ¶ [0039], ¶¶ [0044]-[0046], ¶¶ [0048]-[0054], ¶ [0057], ¶¶ [0069]-[0080], ¶¶ [0082]-[0085], ¶¶ [0088]-[0089], Figs. 2, 5-8.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also Paaske</i> ¶ [0029], ¶ [0031], ¶¶ [0043]-[0044], ¶ [0056], ¶¶ [0059]-[0060], ¶ [0063], ¶ [0101], ¶ [0103], ¶¶ [0110]-[0112], ¶ [0114], ¶ [0124], ¶ [0141], ¶ [0149], ¶ [0188], ¶ [0190], ¶¶ [0217]-[0227], Figs. 2-3. <i>See also Tugenberg</i> at Abstract, 1:40-55, 2:3-30, 2:59-3:49, 3:58-4:9, 4:20-31, 4:39-58, 5:1-9, 5:47-6:2, 6:8-7:67, claims 1-2, 5-8, 10-12, 14-19, Figs. 1-3. <i>See also Aaraj</i> at 8:5-9, 8:13-16, 8:30-31. <i>See Gibod</i> ¶ [0004], ¶ [0008], ¶¶ [0009]-[0010], ¶ [0016], ¶¶ [0021]-[0024], ¶¶ [0026]-[0032], ¶ [0037], ¶ [0046], ¶¶ [0050]-[0051], ¶ [0053], ¶¶ [0069]-[0070], claims 11-14, Fig. 1 <i>See also Mujtaba</i> ¶ [0006], ¶ [0031], ¶ [0033], ¶ [0037], ¶¶ [0042]-[0044], ¶¶ [0052]-[0053], ¶ [0055], ¶ [0057], ¶¶ [0066]-[0068], claims 10, 18, 23, Figs. 1 and 4. <i>See also Buer754</i> at Abstract, 1:42-49, 1:61-2:3, 2:9-12, 2:25-60, 3:42-57, 4:14-60, 5:43-6:12, 6:36-43, 6:53-64, 6:65-7:23, 7:35-41, 8:1-8, 8:35-62, 9:5-12, 9:22-44, 10:23-27, 10:34-37, 10:62-11:3, 11:17-31, 11:62-12:3, 12:47-52, 13:66-14:10, 15:17-33, 16:8-15, 17:1-26, 17:43-62, Figs. 3, 4-6, 8, 11-13, claims 1-3, 6-9, 12, 14, 16. <i>See also Weise</i> Abstract, ¶ [0002], ¶¶ [0005]-[00014], ¶¶ [0021]-[0025], ¶ [0027], ¶ [0030], ¶¶ [0032]-[0043], claims 1, 12-13, 19. <i>See also TLS-I.2</i> at 1, 4, 16, 34, 91-97. <i>See also SSL-3.0</i> at 1, 9, 29, 55-59, 17. <i>See also TLS-I.0</i> at 1, 4, 13, 29-30, 69, 71-73. <i>See also TLS-I.1</i> at 1, 4, 14-15, 31, 74-75, 77-80.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also RFC-5996</i> at 45, 51, 120-123. <i>See also RFC-4306</i> at 88-90, 32. <i>See supra</i> [1a-3], [1c]-[1d].
[1f] store the encrypted session key in the secure memory,	The grounds alleged at the beginning of this chart disclose or suggest the security engine to “store the encrypted session key in the secure memory.” The grounds alleged here uses one of <i>Chou</i>, <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i>’s SoC apparatus to implement <i>Bugiel</i>’s communications method, including to implement IKE (<i>RFC-5996</i> or <i>RFC-4306</i>) or SSL/TLS Handshake Protocol (<i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i>, or <i>TLS-1.2</i>), to establish a secure communication. <i>Bugiel</i> teaches storing a session-token STc (e.g., the claimed “the encrypted session key”) in a secure storage. For example, <i>Bugiel</i> explains “the SSL/TLS session key Sc is generated . . . and sealed (together with pre-master secret PMS and the hash value hash(Certs)). The resulting session-token STc is stored in a secure storage [the claimed “secure memory”] and can be loaded and unsealed later using the session identifier SID.” <i>Bugiel</i> at 225-229, 231, Figs. 2-6. <i>Bugiel</i> also teaches that its method relies on “the TrEE provided by secure hardware is tamper-protected.” <i>Bugiel</i> at 222, 233 (“credentials associated with the SSL/TLS channel (such as pre-master secret key and a session key) are protected by the TrEE”); <i>see also Bugiel</i> at 220-226, 228-236, Fig. 1. Therefore, <i>Bugiel</i> teaches storing the encrypted session key STc in the secure storage accessible only by the secure domain.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	<pre> sequenceDiagram participant MP as Mobile Platform participant WC as Wallet Core participant WH as Wallet Helper participant WS as Web Server Note over MP: Verify Cert_s PMS ← GenRndNm() S_c ← DeriveSessionKey(PMS) ST_c = Seal(hash(Cert_s) S_c PMS) StoreToken(ST_c, SID) PMS_enc ← Enc(PK_s, PMS) Note over WH: 1. Client/Server Hello Note over WS: 2. Cert_s/Server Done Note over WH: 3. SID, Cert_s Note over WH: 4. PMS_enc Note over WS: 5. PMS_enc Note over WS: 6. Change Cipher Note over WH: 7. SID, getFinished Note over WH: 8. ClientFinished Note over WS: 9. Client/Server Finished Note over WS: PMS ← Dec(SK_s, PMS_enc) S_c ← DeriveSessionKey(PMS) </pre> Legend SID = SSL session ID S_c = SSL session key (not known by mWalletHelper) $Cert_s$ = SSL server certificate (SK_s, PK_s) = server key pair PMS = Pre-master secret ST_c = SSL session token Fig. 2. SSL/TLS handshake Further, at the time of the '838 patent, it was well-known to store the encrypted session key in the secure memory, therefore, a POSITA would have found it obvious to store the session-token ST_c in secure memory disclosed in one of <i>Bugiel</i>, <i>Chou</i>, <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i> as discussed in claim feature [1a-2]. <i>See supra</i> [1a-2] (each references' disclosure of protecting sensitive data in their secure memory). The combinations teach this claim feature in three ways: For example, a POSITA would have found it obvious to store the encrypted session key ST_c in the secure memory disclosed in each reference discussed in [1a-2] to protect the encrypted session key as desired by <i>Bugiel</i>. For example, <i>Chou</i> discloses “a secure domain or zone is established by and for the components within HSM 20, in order to set a secure boundary for processing that occurs within this boundary. Security processor 21 provides the various processing within HSM 20, including key authentication and the encrypting and/or decrypting operation to encrypt and/or decrypt data. Secure key module 23 is a storage medium, such as a memory, register, circuit, etc. that stores one or more key value(s) that is used by security processor 21 to encrypt and decrypt data. Secure memory 22 is utilized to store data that is processed and/or to be processed by security processor 21.” <i>Chou</i> ¶ [0021], ¶ [0035] (“The main processing operations for HSM 20 are provided by HSM processor 51, which operates with its BROM 52 and RAM 53.”); <i>see also Chou</i> Abstract, ¶ [0003], ¶ [0005], ¶¶ [0007]-

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	[0009], ¶¶ [0017]-[0019], ¶¶ [0022]-[0028], ¶ [0030], ¶ [0034], ¶¶ [0036]-[0041], ¶¶ [0054]-[0063], ¶¶ [0069]-[0071], Figs. 1 and 3. Therefore, a POSITA would have found it obvious to store the encrypted session key in <i>Chou</i>'s secure memory, e.g., the memories illustrated in Figs 1 and 3 or memories associated with the processing within the security engine, such as security processor, PKA, KEK, etc. Applying the same logic, a POSITA would have also found it obvious to store the encrypted session key in any of the secure memory disclosed in each reference discussed in [1a-2] to protect the encrypted session as desired by <i>Bugiel</i>. See <i>supra</i> [1a-2], [1e]. FIG. 1 FIG. 3 Second, Daedalus appears to argue that the process of encrypting the session key and passing the encrypted session key to the unsecure world involves storing the encrypted session key in a volatile memory. See <i>Infringement Contentions</i>, Ex. F-1 at 21-22. Under Daedalus' view, <i>Chou</i> also discloses feature [1f] in the same manner. <i>Chou</i> explains that “the security processor of the hardware security

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	module wraps context data using the secure key and transfers the wrapped context data for storage in one of the external sources.” <i>Chou</i> at cl. 4; <i>see also id.</i> at cl. 8 (reciting, “the key unit wraps context data using the secure key and transfers the wrapped context data for storage in one of the external sources”), ¶ [0024] (“security processor 21 allows confidential context data to be wrapped using one of the secure keys stored in secure key module 23 and the wrapped context data securely stored in a storage component in the non-secure domain. The operating context data may be a key or it may be data itself. Since the context data is wrapped within HSM 20 prior to the context data leaving HSM.”); <i>see also Chou</i> Abstract, ¶ [0003], ¶ [0005], ¶¶ [0007]-[0009], ¶¶ [0017]-[0019], ¶¶ [0022]-[0028], ¶ [0030], ¶ [0034], ¶¶ [0036]-[0041], ¶¶ [0054]-[0063], ¶¶ [0069]-[0071], Figs. 1 and 3. Thus, under Daedalus’s view of feature [1f], the wrapped/encrypted symmetric session key would be stored in a RAM of <i>Chou</i>’s HSM 20, such as secure memory 22 or HSM RAM 53, etc., when the encrypted key is transferred for storage outside of HSM 20. Further, at the time of the '838 patent, it was well-known to store the encrypted session key in the secure memory. For example, <i>Tkacik</i> teaches moving the plaintext active key 172 (which could be the claimed unencrypted “session key”) to the data register 76 of the security engine 30. <i>Tkacik</i> further teaches encrypting the session key 172 using master key 160 from the key register 60 of the security engine 30, thereby creating a long-lived alternate ciphertext key 372. According to <i>Tkacik</i>, after generating the long-lived alternate ciphertext key 372 (which could be the claimed “encrypted session key”) in the security engine, the encrypted session key 372 is moved from the security engine 30 to the unsecured memories 16, 18, and/or 28 outside the security engine 30. <i>See Tkacik</i> ¶ [0084]. Applying Daedalus’ view, a POSITA would have understood that the ciphertext key 372 is stored, at least temporarily, in a RAM of the secure processing section 30 (e.g., the zeroizable read-write memory 52 or the data register 76, which can itself be part of zeroizable read-write memory 52, <i>see id.</i> ¶ [0054]), when the encrypted session key 372 is generated based on the un-encrypted session key 172 in data register 76. <i>Tkacik</i> explains, for instance, that zeroizable read-write memory 52 performs “data storage and buffering tasks . . . in connection with moving blocks of payload data into and out from secure processing section 30 in support of cryptographic operations.” <i>Id.</i> ¶ [0037]. A POSITA thus would have understood the long-lived alternate ciphertext key 372, which could be the claimed “encrypted session key,” is stored, at least temporarily, in zeroizable read-write memory 52, for

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	example, before transfer to unsecure memory. <i>See id.</i>; <i>see also Tkacik</i> ¶¶ [0001]-[0009], ¶¶ [0019]-[0023], ¶¶ [0028]-[0029], ¶ [0031], ¶ [0034], ¶¶ [0037]-[0039], ¶¶ [0049]-[0058], ¶ [0060], ¶¶ [0064]-[0089], Figs. 2-8, Abstract, claims 1-20; <i>see also supra</i> [1a-2], [1e]. FIG. 2 FIG. 8 In addition, Figures 1 and 2 from <i>Tugenberg</i>, appear below with annotations comparing them to <i>Tkacik</i>'s Figures 2 and 8 above, also annotated (with like colors illustrating like features).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on Bugiel

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	FIG. 1 FIG. 2 <i>Tugenberg</i> explains: “[C]ommunication device 10 contemplates having secret keys stored in non-secure external memory 12 in encrypted form, each secret key having been encrypted with laser-scribed encryption key 21 using secure processing system 14.” <i>Tugenberg</i> at 4:20-25. To encrypt information in this way, and referring to Figure 2, <i>Tugenberg</i> explains: “In step 206, the host processor sends the sensitive information through the encryption logic. In this step, the sensitive information is encrypted with the laser-scribed encryption key by encryption logic circuitry and stored in a memory internal to the secure memory device, and preferably stored in a zeroizable RAM.” <i>Id.</i> at 6:50-55 (emphasis added); see also <i>id.</i> at 2:64-66 (“Encryption logic circuitry 23 preferably implements a symmetric encryption algorithm using laser-scribed encryption key 21.”), 3:34-37 (explaining that “data encrypted by any particular secure memory can only be decrypted by that particular secure memory”). The encrypted information is then transferred to external memory 12. That is, “host processor reads the encrypted sensitive information from the zeroizable memory and stores this information in a memory that is preferably external to the secure memory such as external memory 12 (FIG. 1).” <i>Id.</i> at 6:59-67 (emphasis added); see also <i>id.</i> at Abstract (“A user’s sensitive data is encrypted within the secure memory using the laser-scribed encryption key and stored as encrypted data in the non-secure memory.”), 5:4-5 (“After encryption, the encrypted sensitive information is stored in a non-secure memory.”), claim 12 (stating that “sensitive data is encrypted within the secure memory using the laser-scribed encryption key and stored as encrypted data in the

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	non-secure memory”). <i>See also id.</i> at Abstract, 1:7-10, 1:15-25, 1:40-55, 2:3-3:18, 3:29-4:53, 4:59-5:9, 5:47-6:67, 7:8-8:20, Figs. 1-3, claims 1-2, 5-6, 9 and 12, 14, 17. Thus, although <i>Tugenberg</i>—like <i>Tkacik</i>—teaches to “hav[e] secret keys stored in non-secure external memory 12 in encrypted form,” <i>Tkacik</i> at 4:20-25, <i>Tugenberg</i> also teaches that the encryption process itself involves storing the encrypted secret keys “in a memory internal to the secure memory device,” <i>id.</i> at 6:50-55, at least temporarily, before transferring it to external memory 12. <i>See id.</i> at 4:20-25, 6:50-67. A POSITA would have understood this is how <i>Tkacik</i> operates as well. More specifically, a POSITA would have understood <i>Tugenberg</i>’s teachings above to be synonymous with <i>Tkacik</i>’s teaching that zeroizable read-write memory 52 performs “data storage and buffering tasks . . . in connection with moving blocks of payload data into and out from secure processing section 30 in support of cryptographic operations.” <i>Tkacik</i> ¶ [0037]. As another example, <i>Gibod</i> discloses storing an encrypted key in the secure memory of secure processing domain before exporting it to the open processing domain. <i>See Gibod</i> ¶¶ [0009]-[0010] (“secret information may be exported and/or persisted from the secure processing domain in a secure manner. To export or persist secret information, the secret information may first be encrypted using a persistent (e.g., permanent) encryption key. . . . Once encrypted by the persistent and/or cycling encryption key, the secret information may then be exported from the secure abstraction layer to the client application running in the open processing domain.”), ¶ [0019], ¶ [0048], ¶¶ [0050]-[0051], ¶ [0053], ¶ [0071], claims 11-13, 16, 18, Fig. 3. <i>Gibod</i> also describes “the secure abstraction layer in the secure processing domain.” <i>Gibod</i> Abstract; <i>see also Gibod</i> ¶¶ [0003]-[0009], ¶¶ [0016]-[0017], ¶ [0019], ¶¶ [0023]-[0037], ¶¶ [0040]-[0055], ¶ [0067], ¶¶ [0069]-[0071], claims 1, 5, 11-13, 16, 18, Figs. 1-4. In addition, <i>Buer754</i> also discloses “session keys are encrypted using KEK 238 and stored in a database 240 as security associations” <i>Buer754</i>, 5:43-51. <i>Buer754</i> further describes “[t]heoretically, the encrypted security associations may be stored in any data memory in the network.” <i>Buer754</i>, 5:63-6:3 (emphases added). Therefore, a POSITA would have found it obvious to store the encrypted session key in a secure memory accessible only by the security engine, including e.g., the private keys store 330 accessible only by the security module 320 (part of the claimed “security engine”) in Figure

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	3, EEPROM 336 accessible only by the cryptographic accelerator 326 (part of the claimed “security engine”) in Figure 3, memories 432A-432 B or EEPROM 434 inside cryptographic accelerator 420 (part of the claimed “security engine”) in Figure 4, data and keys storages 532, 534 or 536 of key manager 520 (part of the claimed “security engine”) in Figure 5, etc. In addition, <i>Buer754</i> also teaches storing the encrypted session keys in the security module and cryptographic accelerator (part of the claimed security engine). <i>See Buer754</i> at 6:36-43 (“to protect these keys, the security module 320 encrypts them with a KEK 332 before they are sent to the cryptographic accelerator 326.”); <i>see also</i> 4:14-26, Fig. 4 (encrypted packet & security association sent to cryptographic accelerator 420), Fig. 6 (step 610: encrypt and store session keys), Fig. 11 (steps 1118-1120 : encrypt and store private keys using KEK), Fig. 12 (steps 1210-1212: encrypt session keys using KEK_A and store security associations), 1:42-55, 1:61-2:3, 2:9-13, 2:17-60, 3:42-58, 12:47-52, 4:14-60, 5:43-6:12, 6:4-12, 6:36-43, 6:53-7:64, 8:35-50, 9:5-12, 9:22-44, 10:23-27, 10:62-11:31, 11:62-12:3, 12:30-44, 12:47-52, 12:53-61, 15:12-33, 15:45-16:3, 16:8-20, 16:36-59, 17:1-62; Figs. 1-5, and 13, Abstract, claims 1-3, 6-9, 12, 14, 16. <i>Mujtaba</i> discloses the encrypted session key, e.g., the “recovery blob is generated having embedded therein the session key encrypted by the storage key. . . . Thereafter, the recovery blob . . . are sent to an authorization server.” <i>Mujtaba</i> ¶ [0006], ¶ [0031] (“Storage key 114 and/or the UID (not shown) may be hardwired in the hardware (e.g., burnt fuses) during the manufacturing of wireless processor 101.”), ¶¶ [0042]-[0044], ¶¶ [0052]-[0053] (“a recovery blob is generated by encrypting session key 705 using storage key 114”), ¶¶ [0055]-[0057] (“the recovery blob, is received by the wireless processor”), ¶¶ [0066]-[0068], Figs. 1, 4, 7 and 8, claims 10, 18, 23; <i>see also</i> [1a-1]-[1a-2] (discussing the secure memory within the wireless processor). <i>Paaske</i> also teaches that the TPM encrypts session keys inside TPM’s secure memory before storing the encrypted keys in external flash memory, e.g., TPM keys/data storage 356 of external flash memory 306. <i>See Paaske</i> ¶ [0043] (“[A] key securely stored in the computing system (e.g., in the TPM 202, the integrated circuit 204 or encrypted in the flash memory 206) may be used to decrypt the received information.”), ¶ [0044] (“[T]he TPM 202 may encrypt and/or authenticate information using a secure key before storing the information in the flash memory 206. . . . Again, the key(s) used for the encryption, decryption and authentication operations may be securely stored in the computing system

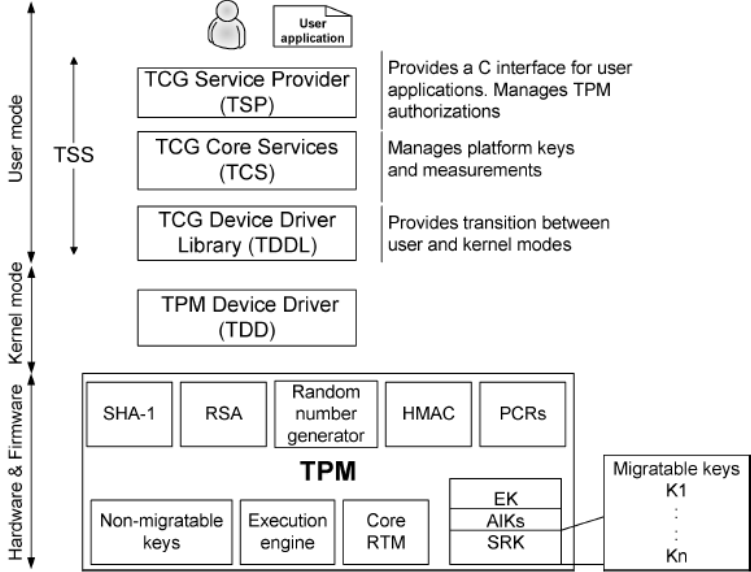
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	(e.g., in the TPM 202, the integrated circuit 204 or encrypted in the flash memory 206).”), ¶ [0056] (“Through the use of these keys and/or keys derived from these keys, the TPM may securely store information outside of the TPM (or the integrated circuit within which the TPM resides). For example, the cryptographic processor(s) 330 may use these keys to encrypt and/or authenticate any data that is stored in an external memory such as the flash memory 306.”), ¶ [0112] (“The TPM (e.g., cryptographic processor(s) 330 in FIG. 3) generates encryption and authentication keys that may be used to encrypt and authenticate information stored in the flash memory. These keys may be securely stored in the flash memory (e.g., in the TPM keys/data section 356) by encrypting the keys using the internal keys discussed above.”), ¶ [0124] (“[T]he system may initialize a 3DES key for decryption operations [T]hese keys may be subsequently encrypted using an internal key and stored in the flash memory (e.g., TPM keys/data 356 in FIG. 3).”). <i>Paaske</i> further explains that the process of writing to external flash memory involves storing the encrypted information temporarily within TPM before transferring to external flash memory through external interface module 322 and flash controller 304. <i>See Paaske</i> ¶¶ [0065]-[0066] (“[T]he cryptographic processor(s) 330 may encrypt the information and generate an authentication digest for the information. Finally, the encrypted information is sent to the flash memory 306 via the external interface module 322 and the flash controller 304.”), ¶ [0048] (“The data buffer 318 provides storage for the processor 308 and other system components. In particular, the data buffer 318 may provide persistent memory, volatile memory, stack/heap space and I/O space for the processor 308.”). In other words, the encrypted session key is stored at least temporarily in a secure memory of the TPM when it is transferred from the TPM to external flash memory. <i>See also Paaske</i> Figs. 2-4, ¶ [0005], ¶¶ [0021]-[0022], ¶¶ [0024]-[0025], ¶ [0029], ¶ [0031]-[0036], ¶¶ [0038]-[0039], ¶¶ [0041]-[0044], ¶¶ [0046]-[0057], ¶¶ [0059]-[0060], ¶ [0063], ¶¶ [0065]-[0068], ¶ [0070], ¶¶ [0101]-[0103], ¶ [0106], ¶ [0108], ¶¶ [0110]-[0114], ¶ [0124], ¶ [0131], ¶ [0133], ¶ [0141], ¶ [0149], ¶¶ [0167]-[0169], ¶ [0188], ¶¶ [0217]-[0227].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	<i>Paaske</i> at Fig. 3. In addition, <i>Aaraj</i> discloses “the endorsement key (EK), is generated by the vendor or manufacturer, and stored in the nonvolatile memory within the TPM. EK should never be disclosed and is only used for . . . creating the storage root key (SRK). . . . The SRK is the root of the hierarchy of TPM keys, and is used for encrypting other migratable keys that are stored outside the TPM.” <i>Aaraj</i> at 8:6. <i>Aaraj</i> also describes migratable keys could include session key, <i>See Aaraj</i> at 8:15 (describing SSL handshake protocol could involve session key exchange between client and server); <i>see also Aaraj</i> at 8:30-31. Fig. 1 of <i>Aaraj</i> shows the migratable keys encrypted with SRK are stored in the secure memory inside the TPM. <i>See Aaraj</i> at 8:5. Therefore, <i>Aaraj</i> teaches this claim feature.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	 Fig. 1. Modules and interfaces of a trusted platform. <i>Aaraj</i> at Fig. 1 Third, in the alleged combinations, <i>Tkacik</i> explicitly teaches storing the encrypted session key 372 in the data register 76 (part of the claimed “secure memory”). <i>Tkacik</i> explains that when the unencrypted session key 172 is needed, a task retrieves a blob that includes the encrypted session key 372 from unsecured memory 16, 18, or 28 and moves it to secure data register 76, then decrypts the blob. <i>See Tkacik</i> ¶ [0073] (explaining that “process 118 may be invoked to recover a plaintext active key 172 which no longer exists within host device 10 but which has an encrypted long-lived alternate ciphertext key 372 counterpart in a blob 110 stored in unsecured memory 16, 18, or 28”), ¶ [0074] (“[A] task 122 retrieves the identified blob 110 from unsecured memory to encryption engine 66, where it is saved in data register 76. . . . [A] task 126 causes encryption processor 66' to decrypt the blob 110 using decryption process 70" of SS algorithm 70 and the BKEK as a cryptographic key. Task 126 recovers plaintext active key 172 from the long-lived alternate ciphertext key 372 portion of the blob 110, which may be moved to active key register

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	72 or otherwise saved within secure processing section 30 for subsequent use.”); <i>see also Tkacik</i> ¶¶ [0065]-[0072], ¶¶ [0075]-[0078], ¶¶ [0083]-[0085], ¶ [0087], Figs. 1-2, 5 and 7. In this way, a POSITA would have understood <i>Tkacik</i>’s security engine stores key 372, which could be the encrypted session key in data register 76 of the secure memory as Figure 2 illustrates below.¹⁵ FIG. 2 <i>Tkacik</i>, Fig. 2 In addition, <i>Tugenberg</i> discloses after the encrypted secret key / common session key is stored in external memory 12, it is retrieved back into secure memory 20 as needed to be decrypted using the laser-scribed encryption key 21 and encryption logic circuit 23. <i>See Tugenberg</i> at 2:5-21 (“When needed by the host processor, the encrypted data is decrypted with encryption logic circuitry within the secure memory and transferred to a zeroizable memory for use by the host processor.”), 4:25-28 (“When needed, communication device decrypts one of the secret keys with encryption logic circuitry

¹⁵ Claim 1 does not require the security engine to “store the encrypted session key in the secure memory” immediately after it “encrypt[s] the symmetric session key in the secure memory.”

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	23 using laser-scribed encryption key 21, and stores the decrypted secret key in zeroizable memory 24.”), 7:14-16 (“Procedure 300 is executed when the host processor requires use the sensitive information that is stored in encryption form.”), 7:26-34 (“[I]n step 304, the host processor reads the encrypted sensitive information, for example from external memory 12 (FIG. 1) and writes this information to the encryption logic circuitry. In step 306, the encryption logic circuitry decrypts the encrypted sensitive information using the laser-scribed encryption key.”), Fig. 3, claim 12 (reciting “retrieving the encrypted sensitive data and an encrypted secret key from the non-secure memory; decrypting, in the secure memory, the encrypted sensitive data and the secret key with the laser-scribed encryption key”). A POSITA thus would have understood the encrypted secret key, which may refer to an encrypted common session key, is stored in encryption logic circuit 23 before decrypting it and storing the decrypted secret key in zeroizable memory 24. <i>See also Tugenberg</i>, Abstract, 2:3-21, 2:59-3:18, 4:20-58, 5:1-9, 6:21-67, 7:8-49, 8:7-20, claims 1-2, 5, 12, Figs. 1 and 3. FIG. 1 FIG. 3 <i>Tugenberg</i>, Figs. 1 and 3.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>Gibod</i> also discloses storing encrypted key in the secure processing domain when loading it from the open processing domain. <i>See Gibod</i> ¶ [0008] (“the secure operation may include, for example, loading a cryptographically wrapped key into the secure abstraction layer”), ¶ [0016], ¶¶ [0025]-[0032], ¶ [0040], ¶ [0046], ¶¶ [0050]-[0051], ¶ [0053], ¶ [0071], claim 5, Fig. 4. Further, <i>Buer754</i> also teaches this claim feature by disclosing sending the encrypted security association to the cryptographic accelerator (part of claimed “security engine”) for decryption. <i>See Buer 754</i>, Fig. 4 (encrypted packet & security association sent to cryptographic accelerator 420), 6:36-43 (“To protect these keys, the security module 320 encrypts them with a KEK 332 before they are sent to the cryptographic accelerator 326”); <i>see also</i>, 1:42-49, 6:4-12, 6:53-7:23, 7:35-41, 7:59-64, 2:9-12, 2:25-49, 12:47-52, 4:14-26, 5:52-6:12, 8:35-50, 9:5-12, 9:22-44, 10:23-27, 10:62-11:3, 11:17-31, 11:62-12:3, 15:17-33, 16:8-15, 16:36-59, 17:1-62; Figs. 3, 5-6, and 11-13, claims 1-3, 6-9, 12, 14, 16. <i>Chou</i> also explains that “the security processor of the hardware security module wraps context data using the secure key and transfers the wrapped context data for storage in one of the external sources.” <i>Chou</i> at claim 4; claim 8 (reciting, “the key unit wraps context data using the secure key and transfers the wrapped context data for storage in one of the external sources”), ¶ [0024] (“security processor 21 allows confidential context data to be wrapped using one of the secure keys stored in secure key module 23 and the wrapped context data securely stored in a storage component in the non-secure domain. The operating context data may be a key or it may be data itself. Since the context data is wrapped within HSM 20 prior to the context data leaving HSM.”); <i>see also Chou</i> Abstract, ¶ [0003], ¶ [0005], ¶¶ [0007]-[0009], ¶¶ [0017]-[0019], ¶¶ [0022]-[0028], ¶ [0030], ¶ [0034], ¶¶ [0036]-[0041], ¶¶ [0054]-[0063], ¶¶ [0069]-[0071], Figs. 1 and 3. Thus, under Daedalus’s view of feature [1f], the wrapped/encrypted symmetric session key would be stored in a RAM of <i>Chou</i>’s HSM 20, such as secure memory 22 or HSM RAM 53, when the encrypted key is transferred for storage outside of HSM 20. <i>Paaske</i> also teach storing the encrypted session key in the TPM’s internal secure storage when loading an encrypted session key from external memory to the TPM. For example, <i>Paaske</i> explains that encrypted keys are loaded into TPM from flash memory to “decrypt the received information.” <i>Paaske</i> ¶ [0043] (“[A] key securely stored in the computing system (e.g., in the TPM 202, the integrated

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	circuit 204 or encrypted in the flash memory 206) may be used to decrypt the received information.”). <i>Paaske</i> further explains that the loaded keys are stored in data buffer 318 (part of the claimed secure memory). <i>See Paaske</i> ¶ [0048] (“[T]he data buffer 318 may provide persistent memory, volatile memory, stack/heap space and I/O space for the processor 308. In some embodiments the persistent memory may be used to stored data (e.g., data 356 and 358 discussed below) that is loaded in from the flash memory 306.”), ¶ [0068] (“The processor 308 (e.g., TPM firmware executing on the processor) reads data stored in the external flash through the external interface module 322 that is attached to the data bus 316.”), ¶ [0133] (describing “reading TPM-related data from the flash memory (e.g., TPM non-volatile ram 358 in FIG. 3) into the persistent memory”), Figs. 2-3. <i>Paaske</i>, Fig. 3

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>Mujtaba</i> also discloses the encrypted session key, e.g., “the recovery blob, is received by the wireless processor” <i>Mujtaba</i> ¶ [0057]; <i>see also Mujtaba</i>, ¶ [0006], ¶ [0031], ¶¶ [0042]-[0044], ¶¶ [0052]-[0053], ¶¶ [0055]-[0057], ¶¶ [0066]-[0068], Figs. 1, 4, 7 and 8, claims 10, 18, 23; <i>see also</i> [1a-1]-[1a-2] (discussing the secure memory within the wireless processor). The SSL/TLS protocols also express the needs to protect the keys used in the secure connection, including the session keys. <i>See TLS-1.2</i> at 96 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”) (emphases added); <i>see also TLS-1.2</i> at 1, 4, 16, 34, 59, 91-97. <i>SSL-3.0</i>, <i>TLS-1.0</i> and <i>TLS-1.1</i> also provide similar disclosures. <i>See SSL-3.0</i> at 59 (“For SSL to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 9, 55-58, 17; <i>TLS-1.0</i> at 73 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 13, 29-30, 44-45, 69, 71-72; <i>TLS-1.1</i> at 80 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 14-15, 31, 48-49, 74-75, 77-80. In addition, the IKE protocol for IPsec also desires protecting the keys. <i>See RFC-5996</i> at 120-123 (Security Considerations); <i>see also RFC-5996</i> at 45, 51. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 88-90 (Security Considerations), 32. As explained for feature [1d], in the combinations based on SSL/TLS protocol, the symmetric session key may be either (1) a “master secret” or (2) a “key_block,” which is used to encrypt/decrypt messages between client and server. In the combinations based on IKE protocol, the symmetric session key <i>S_c</i> could be any of the keys generated in either the initial exchange or Child SA exchange, e.g., <i>g^{ir}</i>, <i>g^{ir}</i> (new), SKEYSEED, SK_d, SK_ai, SK_ar, SK_ei, SK_er, KEYMAT, etc. under Daedalus’s apparent view. <i>See supra</i> [1d].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Exemplary disclosures from the prior art include the following: <i>Bugiel</i> discloses the following at 225 (Establishing SSL/TLS connection and managing two logical SSL/TLS subchannels): Figure 2 illustrates the SSL handshake protocol. The client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters. In the following steps the server's certificate Certs is sent and verified, the SSL/TLS session key Sc is generated (which is never disclosed to WalletHelper) and sealed¹¹ (together with pre-master secret PMS and the hash value hash(Certs)). The resulting session-token STc is stored in a secure storage and can be loaded and unsealed later using the session identifier SID. [Note 8: Sealing means protecting an object so that only a certain set of entities can access or use it.] <i>See also Bugiel</i> at 220-236, Figs. 1-6. <i>See also Chou</i> Abstract, ¶ [0003], ¶ [0005], ¶¶ [0007]-[0009], ¶¶ [0017]-[0019], ¶¶ [0021]-[0028], ¶ [0030], ¶¶ [0034]-[0041], ¶¶ [0054]-[0063], ¶¶ [0069]-[0071], Figs. 1 and 3, claims 4, 8, 19. <i>See also Tkacik</i> ¶¶ [0001]-[0009], ¶¶ [0019]-[0023], ¶¶ [0028]-[0029], ¶ [0031], ¶ [0034], ¶¶ [0037]-[0039], ¶¶ [0049]-[0058], ¶ [0060], ¶¶ [0064]-[0089], Figs. 1-8, Abstract, claims 1-20. <i>See also Tugenberg</i> at Abstract, 1:7-10, 1:15-25, 1:40-55, 2:3-3:18, 3:29-4:53, 4:59-5:9, 5:47-6:67, 7:8-8:20, Figs. 1-3, claims 1-2, 5-6, 9 and 12, 14, 17. <i>See also Gibod</i>, Abstract, ¶¶ [0003]-[0010], ¶¶ [0016]-[0017], ¶ [0019], ¶¶ [0023]-[0037], ¶¶ [0040]-[0055], ¶ [0067], ¶¶ [0069]-[0071], claims 1, 5, 11-13, 16, 18, Figs. 1-4.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also Buer754</i> at 1:42-55, 1:61-2:3, 2:9-13, 2:17-60, 3:42-58, 4:14-60, 5:43-6:12, 6:4-12, 6:36-43, 6:53-7:64, 8:35-50, 9:5-12, 9:22-44, 10:23-27, 10:62-11:31, 11:62-12:3, 12:30-44, 12:47-61, 15:12-33, 15:45-16:3, 16:8-20, 16:36-59, 17:1-62; Figs. 1-6, 11-13, Abstract, claims 1-3, 6-9, 12, 14, 16. <i>See also Mujtaba</i> ¶ [0006], ¶ [0031], ¶¶ [0042]-[0044], ¶¶ [0052]-[0053], ¶¶ [0055]-[0057], ¶¶ [0066]-[0068], Figs. 1, 4, 7 and 8, claims 10, 18, 23. <i>See also Paaske</i> Figs. 2-4, ¶ [0005], ¶¶ [0021]-[0022], ¶¶ [0024]-[0025], ¶ [0029], ¶ [0031]-[0036], ¶¶ [0038]-[0039], ¶¶ [0041]-[0044], ¶¶ [0046]-[0057], ¶¶ [0059]-[0060], ¶ [0063], ¶¶ [0065]-[0068], ¶ [0070], ¶¶ [0101]-[0103], ¶ [0106], ¶ [0108], ¶¶ [0110]-[0114], ¶ [0124], ¶ [0131], ¶ [0133], ¶ [0141], ¶ [0149], ¶¶ [0167]-[0169], ¶ [0188], ¶¶ [0217]-[0227]. <i>See also Aaraj</i> at 8:5, 8:15, 8:30-31, Fig. 1. <i>See also TLS-1.2</i> at 1, 4, 16, 34, 59, 91-97. <i>See SSL-3.0</i> at 1, 9, 55-59, 17. <i>TLS-1.0</i> at 1, 4, 13, 29-30, 44-45, 69, 71-73. <i>TLS-1.1</i> at 1, 4, 14-15, 31, 48-49, 74-75, 77-80. <i>See also RFC-5996</i> at 45, 51, 120-123. <i>See also RFC-4306</i> at 88-90, 32. <i>See supra</i> [1a-2], [1d]-[1e].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
[1g] the system-on-a-chip to establish the secure communication session with the remote server over the network using the session key.	The grounds alleged at the beginning of this chart disclose or suggest “the system-on-a-chip to establish the secure communication session with the remote server over the network using the session key.” The grounds alleged here uses one of <i>Chou</i>, <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i>’s SoC hardware to implement <i>Bugiel</i>’s communications method, including to implement IKE protocol or SSL/TLS Handshake Protocol messages, to establish a secure communication. <i>Bugiel</i> explains that one of its objectives is to provide “[a] secure channel between the user and the web server.” <i>Bugiel</i> at 221. Further, <i>Bugiel</i> teaches applications in which “a user deploys his mobile device to access services provided by remote web servers (over the Internet).” <i>Id.</i> at 221. According to <i>Bugiel</i>, after generating symmetric session key Sc, “[a]ll following messages should be encrypted using Sc.” <i>Bugiel</i> at 225; <i>see also id.</i> at 227 (“TruWalletM establishes the SSL/TLS connection and uses the first negotiated session key for authentication, i.e., the logical SSL/TLS channel is directly established between the WalletCore and WebServer. . . . the logical SSL/TLS channel between WalletCore and WebServer for a secure transmission of the credentials.”). <i>Bugiel</i> thus teaches establishing a secure communication session with a remote server over a network using a session key. <i>See also Bugiel</i> at 220-221, 223-227, 231-232, 234-235. A POSITA would have found it obvious to employ one of <i>Chou</i>, <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i>’s SoC hardware to achieve this. At the time of the '838 patent, it was well known to establish a secure communication session with a remote server over a network using a session key. For example, <i>Chou</i>’s SoC hardware is designed to manage secure keys used to establish secure communications. <i>See Chou</i> ¶ [0006] (“[S]ecure data transfer is utilized to perform secure financial transactions over a network, such as the Internet. . . . The recipient of the secure data utilizes a secure key to decrypt the data to retrieve the financial information.”), ¶ [0069] (“[A] technique for providing secure key operation, in which a secure zone boundary is defined by components that make up a hardware security module (HSM) is described. Secure keys are wrapped/unwrapped and authenticated within the secure zone and do not leave the HSM in an unwrapped state.”). Consistent with the

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	applications <i>Bugiel</i> envisions, <i>Chou</i> teaches applications in which <i>Chou</i>'s SOC 91 (the claimed "system-on-a-chip") "is implemented in a mobile phone 102, such as a cell phone . . . operat[ing] within a cellular network 100." <i>Chou</i> ¶ [0068]; <i>see also id.</i> at Fig. 5. <i>Chou</i> explains: "The various functional blocks of HSM 20, as well as the non-secure components as described above allow HSM 20 to operate within a mobile phone in one embodiment for practicing the invention." <i>Id.</i> ¶¶ [0068]-[0071]; <i>see also id.</i> at ¶¶ [0005]-[0006], ¶¶ [0017]-[0026], ¶¶ [0054]-[0063], ¶¶ [0065]-[0066], ¶¶ [0068]-[0071], Abstract, Figs. 1, 3-5, claims 1, 3, 6-7, 14, 16-18. FIG. 5 Chou, Fig. 5.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	As another example, <i>Tkacik</i> discloses establishing the secure communication session with the remote server by using the plaintext active key 172 as a symmetric cryptographic key for encryption and decryption. <i>See Tkacik</i> ¶ [0056], ¶ [0087]. <i>Tkacik</i> explains: “FIG. 5 shows a flow chart of an exemplary cryptographic operation process 90 performed in secure processing system 12.” <i>Id.</i> ¶ [0064]. Secure processing system 12 is the claimed “system-on-a-chip.” <i>See supra</i> [1pre]; <i>id.</i> ¶ [0025] (explaining components of secure processing system 12 “are all formed together on a common semiconductor substrate and are packaged together and reside within a single integrated circuit 34”). <pre> graph TD 90([CRYPTOGRAPHIC OPERATION]) --> 92{SHORT-LIVED CIPHERTEXT KEY AVAILABLE} 92 -- Y --> 94[LOAD KEY FROM KEY MAGAZINE] 92 -- N --> 116{LONG-LIVED CIPHERTEXT KEY AVAILABLE} 116 -- Y --> 118[LOAD KEY FROM BLOB] 116 -- N --> 138{NEW KEY REQUESTED} 138 -- Y --> 140[MAKE NEW KEY] 138 -- N --> EXIT([EXIT]) 94 --> 166[PERFORM CRYPTOGRAPHIC OPERATION USING ACTIVE KEY] 118 --> 166 140 --> 166 166 --> EXIT </pre> FIG. 5 <i>Tkacik</i>, Fig. 5 Referring to Fig. 5, <i>Tkacik</i> explains that after making or retrieving cryptographic keys (e.g., when processes 94, 118 or 140 completes), the secure processing section 30 of the secure processing systems proceeds to step 166 and performs “cryptographic operation . . . using the plaintext active key 172 as the encryption or decryption key.” <i>Id.</i> ¶ [0087]. Here, the plaintext active key 172 could be the claimed “session key.” A secure communication session is established with a remote server over a

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	network because <i>Tkacik</i> explains that the host device 10, which includes the secure processing system 12 (the claimed “system-on-a-chip”), could be configured for a wide variety of data-processing applications, including “wireless telephone, radio, personal computer, laptop, handheld computer,” etc., to maintain the security of data that may be “communicated by host device 10.” <i>Id.</i> ¶ [0021], ¶ [0080]. Therefore, <i>Tkacik</i> discloses or suggests the system-on-a-chip to establish the secure communication session with the remote server over the network using the session key. <i>See also Tkacik</i> ¶ [0002], ¶¶ [0008]-[0009], ¶¶ [0020]-[0021], ¶¶ [0025]-[0026], ¶¶ [0028]-[0031], ¶ [0034], ¶ [0037], ¶¶ [0052]-[0056], ¶¶ [0064]-[0065], ¶ [0068], ¶¶ [0071]-[0075], ¶¶ [0078]-[0080], ¶¶ [0087]-[0089], Abstract, Figs. 1-2, 5-8, claims 10. As another example, <i>Mujtaba</i> discloses “Distribution server 703 then encrypts data 707 using the session key . . . The session key can then be used [by computing device 701] to decrypt the encrypted data 707.” <i>Mujtaba</i> ¶¶ [0052]-[0053], ¶ [0057] (“the session key is then used to decrypt the distributed data sent from the distribution facility.”); <i>see also Mujtaba</i> ¶ [0006], ¶¶ [0018]-[0020], ¶¶ [0026]-[0027], ¶ [0050]-[0051], ¶¶ [0056]-[0057], ¶¶ [0061]-[0068], Figs. 1-2, 4, 7-8, claims 10, 18, 23. <i>Buer754</i> also discloses SSL provides secure internet transmission between a client and a remote server using the session key. <i>See Buer754</i> at 1:13-49, 1:61-2:13 (security module for protecting keys generated in SSL), 2:24-40 (keys sent to cryptographic accelerator so that the cryptographic accelerator can decrypt the received encrypted data), 5:30-51 (“The cryptographic accelerator 236 decrypts the packets using the session key”), 6:4-35 (describing the cryptographic accelerator 236 uses the session key to encrypt the packet data to send to the network in Fig. 2 and a security module 320 uses the SSL protocol to provide data security in a network in Fig. 3); <i>see also Buer754</i> at Abstract, 3:49-4:26, 5:7-62, 6:53-7:34, 8:63-9:12, 9:16-21, 9:36-44, 10:57-61, 11:3-39, 12:4-23, 12:62-67, 13:1-15:32, 15:65-16:3, 16:15-19, 16:26-59, 17:48-62, Figs 2-6, 12-13, claims 1-9, 14, 16. <i>Tugenberg</i> explains that the host processor (part of the claimed “system-on-a-chip”) “establish[es] a secure session with, for example, an internet vendor. Secure Socket Layer (SSL), for example, is a standard technique used in the internet community to establish such a session.” <i>Tugenberg</i> at 5:50-54, 5:58-61 (“After decryption the host processor sends the decrypted sensitive information to a recipient such as an internet vendor through the secure session previously established.”), 1:52-54 (“What is also

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	needed is a method for communicating sensitive information over a non-secure channel such as the internet.”), 4:54-58 (disclosing “a method of purchasing items over a non-secure communication link, such as the internet, for example using a secure communication device such as communication device 10 of FIG. 1”), 2:36-39 (“Device 10 may, for example, be a wireless communication device, such as a cellular or wireless phone, or a wireline communication device such as a computer, or a portion thereof.”), 4:41-45 (“In any of these embodiments of the present invention, secure data or secure voice or video may be communicated either over wireless or wireline networks.”), Abstract (a “credit card number is encrypted with a session key, and the encrypted credit card number is transferred over the network to a destination such as an internet vendor.”); <i>see also Tugenberg</i> at Abstract, 1:48-54, 2:31-58, 4:20-58, 5:48-61, 8:7-20, claims 1, 5 and 12; <i>see also Gibod</i> ¶ [0071] (TLS); <i>see also Gibod</i> ¶ [0071] (TLS operation), ¶ [0008], ¶ [0016], ¶ [0032], ¶ [0055], ¶¶ [0069]-[0070], Abstract, Figs. 1-2, 5, claims 1, 6-11, 19. In addition, <i>Paaske</i>’s hardware includes components for communication over wireless network. <i>See Paaske</i> ¶ [0032] (“An external interface component 210 may enable the computing system to communicate with external systems via, for example, a wired or wireless network . . . via an external connection 212.”). In addition, <i>Paaske</i>’s hardware is designed to manage secure keys used to establish secure communications. <i>See Paaske</i> ¶ [0043] (“[T]he information received by the TPM 202 may be encrypted by the trusted source In this case, a key securely stored in the computing system (e.g., in the TPM 202, the integrated circuit 204 or encrypted in the flash memory 206) may be used to decrypt the received information.”), ¶ [0031] (“[A] TPM may generate and maintain keys for the computing system. . . . To this end the TPM may include one or more cryptographic processors that perform cryptographic operations including, for example, one or more of encryption, decryption, authentication, verification, signing and key management.”), ¶ [0044] (“[T]he key(s) used for the encryption, decryption and authentication operations may be securely stored in the computing system (e.g., in the TPM 202, the integrated circuit 204 or encrypted in the flash memory 206).”). Consistent with the applications <i>Bugiel</i> envisions, <i>Paaske</i> teaches applications in which <i>Paaske</i>’s integrated circuit chip (the claimed “system-on-a-chip”) may be used in “a mobile device.” <i>Paaske</i> ¶ [0003]; <i>see also id.</i> ¶ [0032] (explaining <i>Paaske</i>’s system may “communicate with external systems via, for example, a wired or wireless network”). A POSITA thus would have found it obvious to implement

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>Bugiel</i>'s teaching to establish secure communications with a remote server over a network using <i>Paaske</i>'s SoC. In addition, <i>Aaraj</i> discloses “once the session key is established, network traffic can be encrypted using the AES_CBC algorithm.” <i>Aaraj</i> at 8:14; <i>see also Aaraj</i> at 8:2, 8:5-8:9, 8:11-8:16, 8:30-8:31. In addition, <i>Weise</i> teaches this claim feature by disclosing its hardware supports SSL and IPsec protocols, which provides secure communications with a remote server. <i>See Weise</i> at ¶ [0035], ¶ [0010], ¶ [0023], ¶ [0033]. <i>Weise</i> further describes its HSM hardware is capable of using the symmetric session key for encrypting data in secure communication session. <i>See Weise</i> ¶ [0005]-[0006], ¶ [0009], ¶ [0013], ¶ [0034]. claims 2, 9, 14, 17, Figs. 3-4. Additionally, in the combinations based on the SSL/TLS protocols (<i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i>, or <i>TLS-1.2</i>), the combinations implement one of the SSL/TLS protocols using security architecture of the alleged combinations, also disclose or suggest the system-on-a-chip to establish the secure communication session with the remote server over the network using the session key. More specifically, each of the SSL/TLS protocols teaches that the SSL/TLS handshake protocol serves to establish secure communication between the client and server. For example, <i>TLS-1.2</i> states that “the TLS Handshake Protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data.” <i>TLS-1.2</i> at 4; <i>see also id.</i> at 6 (“TLS should be used to establish a secure connection between parties.”). <i>TLS-1.2</i> emphasizes that application data can be sent only after the handshake process completes, which establishes the encryption algorithm and keys used for the communication. <i>See id.</i> at 33-37 (Handshake Protocol overview); <i>see also id.</i> at 37-64 (Handshake protocol), 4, 6, 12, 16-19, 24-28, 30, 64-65, 77-82, 91, 94-96. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provide similar disclosures as <i>TLS-1.3</i>. <i>See SSL-3.0</i> at 4 (“the SSL Handshake Protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data SSL should be used to establish a secure connection between two parties.”),

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	18-20 (Handshake Protocol Overview), 20-32 (Handshake Protocol); <i>see also id.</i> at 4-5, 9-11, 16-17, 33-35, 42-46, 50-51, 55, 58-59; <i>see TLS-1.0</i> at 4 (“the TLS Handshake Protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. . . . TLS should be used to establish a secure connection between two parties”), 29-32 (Handshake Protocol overview), 32-47 (Handshake protocol); <i>see also id.</i> at 3-5, 10, 12, 14-17, 21-24, 26, 47-48, 56-60, 62, 69, 72-73; <i>see TLS-1.1</i> at 4-5 (“the TLS Handshake Protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. . . . TLS should be used to establish a secure connection between two parties”), 31-34 (Handshake Protocol Overview), 34-52 (Handshake Protocol); <i>see also id.</i> at 4-6, 11, 13, 15-17, 24-26, 28, 52-53, 63-70, 74-75, 77-80. The SSL/TLS protocols’ disclosure of establishing secure communication with a remote server using the session key is also consistent with the '838 patent, which acknowledges in its background section that SSL protocols, including TLS, are used for establishing secure communication. <i>See</i> '838 patent at 1:10-25 (“One commonly used cryptographic communication protocol is the Secure Sockets Layer (SSL) protocol. <i>The SSL protocol, and the related Transport Layer Security (TLS) protocol, is used for many different types of secure communication sessions</i> including, for example, secure web-browsing, ecommerce, secure upgrades, and other secure communication sessions between two computing devices on an untrusted network such as the Internet”). Additionally, in the combinations based on IKE protocol for IPsec, each IKE protocol (<i>RFC-5996</i> or <i>RFC-4306</i>) also teaches using the system-on-a-chip apparatus in the alleged combinations “to establish the secure communication session with the remote server over the network using the session key.” More specifically, <i>RFC-5996</i> discloses “IKE performs mutual authentication between two parties and establishes an IKE security association (SA) that includes shared secret information that can be used to efficiently establish SAs for Encapsulating Security Payload (ESP) [ESP] or Authentication Header (AH) [AH] and a set of cryptographic algorithms to be used by the SAs to protect the traffic that they carry.” <i>RFC-5996</i> at 5; <i>see also RFC-5996</i> at 1 (“IKE is a component of IPsec used for performing

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	mutual authentication and establishing and maintaining Security Associations (SAs).”), 5-17, 34-40, 44-49, 50-54, 64-65, 75-77, 107, 121-122, 126-131. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 3 (“IKE performs mutual authentication between two parties and establishes and IKE security association (SA) that includes shared secret information that can be used to efficiently establish SAs for Encapsulating Security Payload (ESP) [RFC4303] and/or Authentication Header (AH) [RFC4302] and a set of cryptographic algorithms to be used by the SAs to protect the traffic that they carry.”), 1, 3-13, 21-24, 27-34, 77-78, 88-95. In addition, as discussed in claim features [1b]-[1d], each of the IKE protocols teaches generating, based on the cryptographic key exchange, a symmetric session key to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session. For example, <i>RFC-5996</i> discloses generating a series of shared keys, such as g^{ir}, g^{ir} (new), SKEYSEED, SK_d, SK_ai, SK_ar, SK_ei, SK_er, KEYMAT, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session. <i>See RFC-5996</i> at 3, 5-6, 9-23, 25-26, 30-33, 39-41, 44-48, 50-54, 72, 74, 80-83, 86-87, 119-121, 132-137; <i>see also RFC-4306</i> at 3-12, 14, 19-24, 26-34, 42-44, 47, 50-52, 55-56, 64, 86, 88-89, 96-97. Therefore, each of the IKE protocols teaches establishing the secure communication with the remote server using the session key. Exemplary disclosures from the prior art include the following: <i>Bugiel</i> discloses the following at 220: <i>Contribution.</i> . . . To the best of our knowledge, no other solution exists for mobile phones that meets these requirements. Our key design solution, which allows us to meet these requirements, is to split a single SSL/TLS connection between the user device and the server into two logically separated channels, where one is protected by TrEE and is used to transmit passwords, and another one is intended for conventional data. Our design relies on the

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	availability of a Trusted Execution Environment (TrEE) to protect the execution of critical code from tampering. Such TrEEs can be provided by commodity secure hardware for mobile devices such as M-Shield [8] and ARM TrustZone [9]. <i>See also Bugiel</i> at 220-221, 223-227, 231-232, 234-236. <i>See also Chou</i> at ¶¶ [0005]-[0006], ¶¶ [0017]-[0026], ¶¶ [0054]-[0063], ¶¶ [0065]-[0066], ¶¶ [0068]-[0071], Abstract, Figs. 1, 3-5, claims 1, 3, 6-7, 14, 16-18. <i>See also Tkacik</i> ¶ [0002], ¶¶ [0008]-[0009], ¶¶ [0020]-[0021], ¶¶ [0025]-[0026], ¶¶ [0028]-[0031], ¶ [0034], ¶ [0037], ¶¶ [0052]-[0056], ¶¶ [0064]-[0065], ¶ [0068], ¶¶ [0071]-[0075], ¶¶ [0078]-[0080], ¶¶ [0087]-[0089], Abstract, Figs. 1-2, 5-8, claims 10. <i>See also Mujtaba</i> ¶ [0006], ¶¶ [0018]-[0020], ¶¶ [0026]-[0027], ¶ [0050]-[0053], ¶¶ [0056]-[0057], ¶¶ [0061]-[0068], Figs. 1-2, 4, 7-8, claims 10, 18, 23. <i>See also Buer754</i> at Abstract, 1:13-49, 1:61-2:13, 2:24-40, 3:49-4:26, 5:7-62, 6:4-35, 6:53-7:34, 8:63-9:12, 9:16-21, 9:36-44, 10:57-61, 11:3-39, 12:4-23, 12:62-67, 13:1-15:32, 15:65-16:3, 16:15-19, 16:26-59, 17:48-62, Figs 2-6, 12-13, claims 1-9, 14, 16. <i>See also Tugenberg</i> at Abstract, 1:48-54, 2:31-58, 4:20-58, 5:48-61, 8:7-20, claims 1, 5 and 12. <i>See also Gibod</i> ¶ [0008], ¶ [0016], ¶ [0032], ¶ [0055], ¶¶ [0069]-[0071], Abstract, Figs. 1-2, 5, claims 1, 6-11, 19. <i>See also Paaske</i> ¶ [0003], ¶ [0006], ¶ [0028], ¶¶ [0030]-[0032], ¶ [0041], ¶¶ [0043]-[0044], ¶ [0055], ¶ [0064], ¶ [0101], ¶ [0106], ¶¶ [0110]-[0112], ¶ [0124], ¶ [0169], ¶¶ [0184]-[0185], ¶¶ [0188]-[0189], ¶ [0222], Fig. 2. <i>See also Aaraj</i> at 8:2, 8:5-8:9, 8:11-8:16, 8:30-8:31.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also Weise</i> at ¶¶ [0005]-[0010], ¶ [0013], ¶ [0023], ¶ [0029], ¶¶ [0032]-[0035], ¶ [0023], ¶¶ [0042]-[0043], claims 2, 9, 14, 17, Figs. 3-4. <i>See also TLS-1.2</i> at 4, 6, 12, 16-19, 24-28, 30, 33-65, 77-82, 91, 94-96. <i>See also SSL-3.0</i> at 4-5, 9-11, 16-35, 42-46, 50-51, 55, 58-59. <i>See also TLS-1.0</i> at 3-5, 10, 12, 14-17, 21-24, 26, 29-48, 56-60, 62, 69, 72-73. <i>See also TLS-1.1</i> at 4-6, 11, 13, 15-17, 24-26, 28, 31-53, 63-70, 74-75, 77-80. <i>See also RFC-5996</i> at 1, 3, 5-23, 25-26, 30-41, 44-49, 50-54, 64-65, 72, 74-77, 80-83, 86-87, 107, 119-122, 126-137. <i>See also RFC-4306</i> at 1, 3-14, 19-24, 26-34, 42-44, 47, 50-52, 55-56, 64, 77-78, 86, 88-97. <i>See also supra</i> [1pre], [1b]-[1d].
Claim 2	
[2] The system-on-a-chip apparatus of claim 1, wherein the security engine is further to receive a server nonce from the remote server in a response message to the request for the secure communication session, the response message including a server nonce	The grounds alleged at the beginning of this chart disclose or suggest “the security engine is further to receive a server nonce from the remote server in a response message to the request for the secure communication session, the response message including a server nonce and store the server nonce in the secure memory.” Figure 2 of <i>Bugiel</i> illustrates exchanging Client/Server Hello described in SSL/TLS handshake and specifically refers to <i>TLS-1.2</i> for more details regarding TLS/SSL handshake messages, including the ServerHello message. <i>See Bugiel</i> at 225 (“All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16].” (citing <i>TLS-1.2</i>)).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
and store the server nonce in the secure memory.	<pre> sequenceDiagram participant MP as Mobile Platform participant WC as Wallet Core participant WH as Wallet Helper participant WS as Web Server Note over WH: 1. Client/Server Hello WH->>WS: 2. Cert_s/Server Done WS->>WH: 3. SID, Cert_s WH->>WC: 4. PMS_enc Note over WC: Verify Cert_s PMS ← GenRndNm() S_c ← DeriveSessionKey(PMS) ST_c = Seal(hash(Cert_s) S_c PMS) StoreToken(ST_c, SID) PMS_enc ← Enc(PK_s, PMS) Note over WS: 5. PMS_enc WS->>WH: 6. Change Cipher Note over WH: 7. SID, getFinished WH->>WC: 8. ClientFinished Note over WC: ST_c ← LoadToken(SID) hash(Cert_s) S_c PMS ← Unseal(ST_c) ClientFinished ← getFinished(S_c) Note over WS: 9. Client/Server Finished Note over WS: PMS ← Dec(SK_s, PMS_enc) S_c ← DeriveSessionKey(PMS) </pre> Legend SID = SSL session ID S_c = SSL session key (not known by mWalletHelper) $Cert_s$ = SSL server certificate (SK_s, PK_s) = server key pair PMS = Pre-master secret ST_c = SSL session token Fig. 2. SSL/TLS handshake Each of the SSL/TLS protocols (<i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i> and <i>TLS-1.2</i>) explains the SSL/TLS Handshake Protocol describes receiving a server nonce from the remote server in a response message from the client requesting the secure communication session, the response message including a server nonce, and storing the server nonce in the secure memory. The combinations based on one of the TLS/SSL protocols (<i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i>, or <i>TLS-1.2</i>) alleged here uses one of <i>Chou</i>, <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i>'s hardware to implement <i>Bugiel</i>'s communications method, including to implement SSL/TLS Handshake Protocol messages, to establish a secure communication. <i>See supra</i> [1pre], [1a-1]-[1a-2]. According to <i>TLS-1.2</i>, the client receives a ServerHello message (the claimed “response message”), including ServerHello.random (the claimed “server nonce”), according to the SSL/TLS Handshake protocol. More specifically, <i>TLS-1.2</i> explains that, in the TLS Handshake Protocol, the first and required step is for a client and server to exchange two messages: ClientHello (the claimed “request for the secure communication session”) and ServerHello (the claimed “a response message to the request”), where ServerHello includes ServerHello.random (the claimed “server nonce”). <i>See TLS-1.2</i>

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	at 33-34 (“The TLS Handshake Protocol involves the following steps: Exchange hello messages . . . Additionally, two random values are generated and exchanged: ClientHello.random and ServerHello.random”), 42-43 (“The server will send this message [ServerHello message] in response to a client hello message”), Figs. 1-2; <i>see also</i> <i>TLS-1.2</i> at 38-47 (Hello Messages), 17-18, 20, 26, 33-38, 52, 55, 64, 70-71, 73, 75, 77, 85, 87-88, 92, 94, 100.  <pre> sequenceDiagram participant Client participant Server Client->>Server: ClientHello Server-->>Client: ServerHello Server-->>Client: [ChangeCipherSpec] Server-->>Client: Finished Client-->>Server: [ChangeCipherSpec] Client-->>Server: Finished Client-->>Server: Application Data Server-->>Client: Application Data </pre> Figure 2. Message flow for an abbreviated handshake <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provides similar disclosures as <i>TLS-1.2</i>. <i>See</i> <i>SSL-3.0</i> at 18 (“These processes are performed in the handshake protocol, which can be summarized as follows: The client sends a client hello message to which the server must respond with a server hello message, or else a fatal error will occur and the connection will fail. . . . Additionally, two random values are generated and exchanged: ClientHello.random and ServerHello.random.”), 24 (“The server processes the client hello message and responds with . . . server hello message.”); <i>see also</i> <i>SSL-3.0</i> at 18-28, 32-35, 37-38, 41, 49, 51, 56-59; <i>TLS-1.0</i> at 29-30 (“The TLS Handshake Protocol involves the following steps: Exchange hello messages . . . exchange random values, . . . These goals are achieved by the handshake protocol, which can be summarized as follows: The client sends a client hello message to which the server must respond with a server hello message, or else a fatal error will occur and the connection will fail. . . . Additionally, two random values are generated and exchanged: ClientHello.random and ServerHello.random”), 36 (“The server will send this message [ServerHello message] in response to a client hello message”), Figs. 1-2; <i>see also</i> <i>TLS-1.0</i> at 14-15, 21-23, 29-42, 47, 51-52, 54, 56, 64, 66, 70-72; <i>TLS-1.1</i> at 31-32 (“The TLS Handshake Protocol involves the following steps: Exchange hello

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	messages . . . exchange random values, . . . These goals are achieved by the handshake protocol, which can be summarized as follows: The client sends a client hello message to which the server must respond with a server hello message, or else a fatal error will occur and the connection will fail. . . Additionally, two random values are generated and exchanged: ClientHello.random and ServerHello.random”), 39 (“The server will send this message [ServerHello message] in response to a client hello message”), Figs. 1-2; <i>see also TLS-1.1</i> at 16, 24, 31-46, 52-53, 57-58, 60, 64, 70-71, 76, 78, 83. As discussed in [1b], ClientHello.random and ServerHello.random are random nonces exchanged between client and remote server to avoid “replay” attack in SSL/TLS protocols. <i>See TLS-1.2</i> at 91-92; <i>SSL-3.0</i> at 55, 57; <i>TLS-1.0</i> at 69-71; <i>TLS-1.1</i> at 74-76.¹⁶ In addition, <i>TLS-1.0</i> incorporates by reference the original SSL patent <i>Elgamal</i>. <i>See TLS-1.0</i> at 74. <i>Elgamal</i> also teaches including “a random number used to ensure channel integrity” in the client-hello message, which the server responds with a server-

¹⁶ Before the earliest possible priority date of the '838 patent, many patent documents describe the random numbers exchanged between the server and client as random nonce. *See* U.S. Patent No. 6,643,774 to McGarvey (e.g., at 7:16-8:3), which was filed on April 8, 1999 and patented on November 4, 2003; U.S. Patent No. 7,039,946 to Binding et al. (e.g., at 10:41-57, 11:52-12:4, 14:52-15:6), which was filed on October 12, 1999 and patented on May 2, 2006; U.S. Patent No. 6,775,772 to Binding et al. (e.g., at 9:64-10:13, 11:5-23, 14:1-21), which was filed on October 12, 1999 and patented on August 10, 2004; U.S. Patent No. 6,694,431 to Binding et al. (e.g., at 10:10-26, 11:18-37, 14:15-35), which was filed on October 12, 1999 and patented on February 17, 2004; U.S. Patent No. 6,751,731 to Binding et al. (e.g., 9:63-10:12, 11:12-18, 14:15-35), which was filed on October 12, 1999 and patented on June 15, 2004; U.S. Patent Application Publication No. 2010/0205443 to Zhao et al. (e.g., at ¶ [0114], ¶¶ [0175]-[0177], ¶ [0186]), which was filed on April 23, 2010 and published on August 12, 2010; U.S. Patent Application Publication No. 2011/0231653 to Bollay et al., (e.g., at ¶¶ [0003]-[0005], ¶¶ [0022]-[0029], ¶ [0036], ¶ [0062], ¶ [0070], ¶ [0090], ¶ [0137]), which was filed on December 13, 2010 and published on September 22, 2011; U.S. Patent No. 5,657,390 to Elgamal et al (e.g., at 6:35-55, 33:24-37, Figs. 9-11), which was filed on August 25, 1995 and patented on August 12, 1997; U.S. Patent Application Publication No. 2010/0042824 to Lee et al. (e.g., at ¶ [0074], ¶ [0081], ¶¶ [0087]-[0088], ¶ [0118], ¶ [0130]), which was filed on August 14, 2009 and published on February 18, 2010; *Buer-754* at 1:36-41, 1:61-2:8, 6:13-31, 13:21-37, 13:45-65, 14:11-34, 14:43-55; EP1706956B1 to Cam-Winget et al. (e.g., at ¶¶ [0139]-[0142]), which was filed on January 10, 2005 and published on October , 2006; U.S. Patent Application Publication No. 2004/0005061 to Buer et al. (e.g., at ¶ [0005], ¶¶ [0009]-[0010], ¶¶ [0050]-[0051], ¶¶ [0108]-[0115], ¶¶ [0117]-[0124], ¶¶ [0127]-[0134], ¶¶ [0136]-[0140]), which was filed on July 8, 2002 and published on January 8, 2004.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	hello message including a server random number. <i>Elgamal</i> at 7:13-40; <i>see also id.</i> at Figs. 4-7. <i>Elgamal</i> further describes the random number as “nonce” because they are “used to defeat ‘playback’ attacks.” <i>Elgamal</i> at 33:24-31. <i>See, e.g., Callaway Golf Co. v. Acushnet Co.</i>, 576 F.3d 1331, 1346 (Fed. Cir. 2009) (“To incorporate matter by reference, a host document must contain language ‘clearly identifying the subject matter which is incorporated and where it is to be found.’” (quoting <i>In re De Seversky</i>, 474 F.2d 671, 674 (CCPA 1973))) Therefore, the combinations based on SSL/TLS protocols disclose or suggest the first part of claim 2: “the security engine is further to receive a server nonce from the remote server in a response message to the request for the secure communication session, the response message including a server nonce.” In addition, the combinations based on SSL/TLS also disclose or suggest the second part of claim 2: “store the server nonce in the secure memory.” Additionally, each SSL/TLS protocol teaches that a ServerHello.random (or server nonce) is one of the essential security parameters for calculating shared secrets and keys. More specifically, each SSL/TLS protocol explains the variable server_random (i.e., the server random nonce) is part of the SecurityParameters structure. <i>See TLS-1.2</i> at 18, 77; <i>see also SSL-3.0</i> at 21-25; <i>TLS-1.0</i> at 15, 56; <i>TLS-1.1</i> at 16, 64. <i>TLS-1.2</i> also teaches using the server nonce to generate a master key, and an encryption key_block, etc. <i>See TLS-1.2</i> at 26, 34, 64, 85, 92, 94; <i>see also SSL-3.0</i> at 26, 32-35, 58-59; <i>TLS-1.0</i> at 21-23, 29, 41, 47, 72; <i>TLS-1.1</i> at 24, 31, 44, 52-53, 78; <i>see supra</i> claim features [1c]-[1d]. <i>TLS-1.2</i> explains: “For TLS to be able to provide a secure connection, . . . keys . . . must be secure.” <i>TLS-1.2</i> at 96. <i>see also TLS-1.2</i> at 1, 4, 16, 34, 59, 85, 91-97. <i>SSL-3.0</i>, <i>TLS-1.0</i> and <i>TLS-1.1</i> also provide similar disclosures. <i>See SSL-3.0</i> at 59 (“For SSL to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 9, 55-58, 17; <i>TLS-1.0</i> at 73 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 13, 29-30, 69, 71-72; <i>TLS-1.1</i> at 80 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 14-15, 31, 74-75, 77-80. With the information of the random nonces, an attacker would be able to derive the keys based on the random nonces, making the keys unsecured. The SSL/TLS protocols are based on the premise that “attackers have substantial computational resources and cannot obtain secret information from sources outside the protocol.” <i>TLS-1.2</i> at 91 (Security Analysis); <i>see also</i> <i>SSL-3.0</i> at 55; <i>TLS-1.0</i> at 69; <i>TLS-1.1</i> at 74. Keeping the random nonce in secure memory helps ensure this assumption stays true and that attackers cannot obtain secret information, such as the random nonce, by accessing the insecure memory. Storing the server nonce in insecure memory would, contrary to the basic premise of TLS/SSL protocol, allow an attacker to obtain the server nonce from insecure memory. A POSITA thus would have understood that data to be processed securely, including the TLS/SSL server nonce, would be stored in secure memory disclosed in the combinations when implementing <i>Bugiel</i>’s TruWalletM process. <i>See supra</i> [1a-1]-[1a-2], [1f] (disclosures of secure execution in the security engine and secure memory in each reference); <i>see, e.g.,</i> <i>Chou</i> ¶¶ [0020]-[0021] (showing a secure zone 80 is established for the components within HSM 20, whereas components 11-14 outside that boundary are in a non-secure zone). <i>Chou</i>’s Figure 1 illustrates a secure memory 22 “to store data that is processed and/or to be processed by security processor 21.” <i>Chou</i> ¶ [0021]. Thus, to operate on the server nonce to calculate other shared secrets and keys securely, the server random nonce must be stored in a secure memory to be protected from attackers. <i>See TLS-1.2</i> at 25-26 (Key Calculation), 64 (Computing the Master Secret); <i>see also</i> <i>TLS-1.2</i> at 34, 94; <i>see also</i> <i>SSL-3.0</i> at 26, 32-35, 58-59; <i>TLS-1.0</i> at 21-23, 29, 41, 47, 72; <i>TLS-1.1</i> at 24, 31, 44, 52-53, 78; <i>see supra</i> claim features [1c]-[1d]. In this way, the grounds alleged here also teach the second part of claim 2: “store the server nonce in the secure memory.”

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>										
	Additionally, the combinations based on IKE protocols (<i>RFC-5996</i> or <i>RFC-4306</i>) alleged here also teach claim 2: “to receive a server nonce from the remote server in a response message to the request for the secure communication session, the response message including a server nonce and store the server nonce in the secure memory.” For example, <i>RFC-5996</i> discloses in response to the initiator’s message, the responder “sends its nonce in the Nr payload” (the claimed “server nonce”) in both the initial exchange and CREATE_CHILD_SA exchange. <i>RFC-5996</i> at 10-11; <i>see also RFC-5996</i> at 6, 9-11, 13-16, 31, 33, 37-39, 44, 46-48, 50, 52-53, 62, 69, 74, 96, 121-122, 134, 137. The initial exchanges are as follows: <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; text-align: left;">Initiator</td> <td style="width: 50%; text-align: right;">Responder</td> </tr> <tr> <td colspan="2" style="text-align: center;">-----</td> </tr> <tr> <td colspan="2" style="padding-top: 10px;">HDR, SAi1, KEi, Ni --></td> </tr> <tr> <td colspan="2" style="padding-top: 20px;">HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. The SAi1 payload states the cryptographic algorithms the initiator supports for the IKE SA. The KE payload sends the initiator’s Diffie-Hellman value. Ni is the initiator’s nonce.</td> </tr> <tr> <td colspan="2" style="text-align: right; padding-top: 20px;"><-- HDR, SAR1, KEr, Nr, [CERTREQ]</td> </tr> </table> <i>RFC-5996</i> at 10. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 8 (“The responder chooses a cryptographic suite from the initiator’s offered choices and expresses that choice in the SAR1 payload, completes the Diffie-Hellman exchange with the Ker payload, and sends its nonce in the Nr payload.”), 4, 7, 10, 19, 23, 26, 28-30, 32-34, 39, 45, 64, 89. Therefore, the combinations based on IKE protocols satisfy the first part of claim 2. Each IKE protocol also discloses the need to keep the nonces secure for further computations. <i>See RFC-5996</i> at 44 (Nonces), 120-123 (Security Considerations), 46-47, 52-53; <i>RFC-4306</i> at 26	Initiator	Responder	-----		HDR, SAi1, KEi, Ni -->		HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. The SAi1 payload states the cryptographic algorithms the initiator supports for the IKE SA. The KE payload sends the initiator’s Diffie-Hellman value. Ni is the initiator’s nonce.		<-- HDR, SAR1, KEr, Nr, [CERTREQ]	
Initiator	Responder										

HDR, SAi1, KEi, Ni -->											
HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. The SAi1 payload states the cryptographic algorithms the initiator supports for the IKE SA. The KE payload sends the initiator’s Diffie-Hellman value. Ni is the initiator’s nonce.											
<-- HDR, SAR1, KEr, Nr, [CERTREQ]											

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

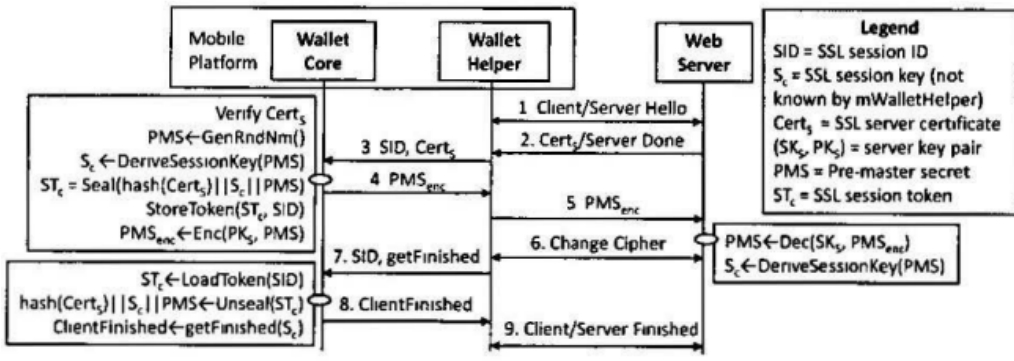
The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	(Nonces), 88 (Security Considerations), 28-29, 33. Thus, to operate on the server nonce to calculate other shared secrets and keys securely, the server random nonce must be stored in a secure memory to be protected from attackers. As discussed in [1a-1]-[1a-2] and [1f], the combinations based on <i>Bugiel</i> teach a secure memory that is only accessible by the security engine. <i>See supra</i> [1a-1]-[1a-2], [1f] (disclosures of secure memory and security engine for every prior art reference). Therefore, the server nonce also needs to be stored in the secure memory within the secure processing section for calculation. Therefore, the grounds alleged here teach claim 2. Exemplary disclosures include the following: <i>Bugiel</i> discloses the following at 225-227 (Establishing SSL/TLS connection and managing two logical SSL/TLS subchannels): Figure 2 illustrates the SSL handshake protocol. The client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters. In the following steps the server's certificate Certs is sent and verified, the SSL/TLS session key Sc is generated (which is never disclosed to WalletHelper) and sealed (together with pre-master secret PMS and the hash value hash(Certs)). The resulting session-token STc is stored in a secure storage and can be loaded and unsealed later using the session identifier SID. ... All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16]. <i>See also Bugiel</i> at 235-236. <i>TLS-1.2</i> discloses the following at 42-43 (Server hello): When this message will be sent:

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	The server will send this message in response to a client hello message when it was able to find an acceptable set of algorithms. If it cannot find such a match, it will respond with a handshake failure alert. Structure of this message: <pre> struct { ProtocolVersion server_version; Random random; SessionID session_id; CipherSuite cipher_suite; CompressionMethod compression_method; } ServerHello; ... random </pre> This structure is generated by the server and MUST be independently generated from the ClientHello.random. <i>See also TLS-1.2</i> at 1, 4, 16-18, 20, 25-26, 33-47, 52, 55, 59, 64, 70-71, 73, 75, 77, 85, 87-88, 91-97, 100. <i>See also SSL-3.0</i> at 1, 9, 17-28, 32-35, 37-38, 41, 49, 51, 55-59. <i>See also TLS-1.0</i> at 1, 4, 13-15, 21-23, 29-42, 47, 51-52, 54, 56, 64, 66, 69-74. <i>See also TLS-1.1</i> at 1, 4, 14-16, 24, 31-46, 52-53, 57-58, 60, 64, 70-71, 74-80, 83. <i>See also RFC-5996</i> at 6, 9-11, 13-16, 31, 33, 37-39, 44, 46-48, 50, 52-53, 62, 69, 74, 96, 120-123, 134, 137. <i>See also RFC-4306</i> at 4, 7-8, 10, 19, 23, 26, 28-30, 32-34, 39, 45, 64, 88-89. <i>See supra</i> claim 1.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
Claim 3	
[3] The system-on-a-chip apparatus of claim 1, wherein the security engine is further to generate a pre-master key, encrypt the pre-master key using the security key of the security engine, and send the encrypted pre-master key to the remote server.	The grounds alleged at the beginning of this chart disclose or suggest “the security engine is further to generate a pre-master key, encrypt the pre-master key using the security key of the security engine, and send the encrypted pre-master key to the remote server.” As an initial matter, the '838 patent's specification does not support claim 3. The '838 patent only discloses encrypting the pre-master key using <i>a server public key provided by the server</i> before sending the encrypted pre-master key to the server. <i>See</i> '838 pat. at 8:28-34, 10:60-64. The '838 patent does not describe encrypting the pre-master key <i>using the security key of the security engine, which is encoded to the secure memory of the security engine during manufacturing</i>. Patent Owner may argue claim 3 is supported by the following description: “The keys and related keying data generated during the client key exchange may be stored in the secure memory 114 in an encrypted state (using the security key 150) or an unencrypted state.” '838 pat. at 11:6-9. However, this description, at most, only describes encrypting the pre-master key using the security key 150 for storage purpose, not for sending to the remote server. In any event, to the extent that Daedalus argues that claim 3 is supported by the specification of the '838 patent, including (1) “the security engine 110 generates the pre-master key;” (2) “encrypts the pre-master key using the server public key received in block 410 prior to sending the encrypted pre-master key to the server;” and (3) “[t]he keys and related keying data generated during the client key exchange may be stored in the secure memory 114 in an encrypted state (using the security key 150)” ('838 pat. at 8:23-34, 10:60-64, 11:6-9), the grounds alleged here provide the same disclosures and thus would satisfy claim 3. First, like the specification of the '838 patent (<i>see</i> '838 pat. at 8:23-34, 10:60-64), the grounds alleged here include a security engine that generates a pre-master secret (i.e., the claimed “pre-master key”).

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	<i>Bugiel</i> teaches generating a PMS, i.e., pre-master secret (the claimed “pre-master key”) during the cryptographic key exchange. <i>See Bugiel</i> at Fig. 2; <i>see also</i> 225-230, 233  Fig. 2. SSL/TLS handshake The grounds alleged here uses the security engine as discussed in claim feature [1a-1] to implement <i>Bugiel</i>’s teaching of generating a pre-master secret in the secure execution environment. For example, <i>Chou</i>’s HSM 20 (part of the claimed “security engine”) could be used to implement <i>Bugiel</i>’s teaching of generating a pre-master secret. In particular, <i>Chou</i> explains that PKA module 56 (part of HSM 20 and thus part of the claimed “security engine”) performs cryptographic key exchange algorithms, such as Diffie-Hellman shared secret computation or RSA key generation. <i>See Chou</i> ¶¶ [0043]-[0053], ¶¶ [0054]-[0055]; <i>see also</i> ’838 pat. at Abstract (“The cryptographic key exchange may be, for example, a Rivest-Shamir-Adleman (RSA) key exchange or a Diffie-Hellman key exchange”), 8:23-26 (“wherein an RSA key exchange is used, the security engine . . . generates a random pre-master key.”). In addition, the SoC apparatus (including the security engine) disclosed in other combinations could also be used to implement <i>Bugiel</i>’s teaching as discussed at the beginning of this chart. <i>See e.g.</i>, <i>Paaske</i> ¶ [0055], ¶ [0106], ¶ [0169], ¶ [0194], ¶¶ [0205]-[0206], ¶¶ [0208]-[0210]; <i>see Tkacik</i> ¶¶ [0079]-[0080]; <i>Tugenberg</i> at 5:47-54; <i>Mujtaba</i> ¶ [0006], ¶¶ [0052]-[0053], ¶¶ [0055]-[0057]; <i>Buer754</i> at 1:27-41, 1:61-2:8, 5:24-29, 6:13-31, 9:16-21, 13:14-16:35; <i>Gibod</i> ¶ [0037], ¶ [0071], Fig. 1; <i>Aaraj</i>

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	at 8:1, 8:3-4, 8:6-11, 8:15, 8:31; <i>Weise</i> ¶ [0010], ¶ [0023], ¶¶ [0033]-[0035], Figs. 3 and 5, claims 9 and 17; <i>see supra</i> [1pre]-[1a-1], [1c]-[1d]. Additionally, each SSL/TLS protocol explains that one of the keys generated using SSL/TLS protocols is the pre-master secret (the claimed “pre-master key”). <i>See TLS-1.2</i> at 91 (“The general goal of the key exchange process is to create a pre_master_secret known to the communicating parties and not to attackers. The pre_master_secret will be used to generate the master_secret . . . required to generate the . . . encryption keys . . .”), 58-61 (RSA encrypted premaster secret message); <i>see also TLS-1.2</i> at 5, 34, 50-51, 53, 56-60, 64-65, 74, 86-87, 91-93. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provide related disclosures. <i>See SSL-3.0</i> at 28-30, 32-33, 40, 51, 55-57; <i>TLS-1.0</i> at 29, 39-40, 43-45, 47-48, 53, 55, 69-71, 74; <i>TLS-1.1</i> at 31, 42, 47-50, 52-53, 59-60, 75-76. Therefore, in the grounds alleged here, the security engine as discussed in [1a-1] generates a TLS/SSL pre-master secret (i.e., the claimed “pre-master key”) based on key exchange. Second, like the '838 specification (<i>see</i> '838 pat. at 8:28-34, 10:60-64), the grounds alleged here includes encrypting the pre-master key using a server public key before sending the encrypted pre-master key to the server. For example, <i>Bugiel</i>'s Figure 2 teaches encrypting the pre-master key using its Wallet Core (part of the security engine in the secure execution environment) when it discloses sending the encrypted pre-master key PMS_{enc} to the Web Server. <i>See Bugiel</i> at Fig. 2 (“PMS_{enc}←(PKs, PMS),” where PKs is the server public key). Additionally, each SSL/TLS protocol also teaches encrypting a premaster secret using the server public key and sending the encrypted premaster secret to the remote server. For example, <i>TLS-1.2</i> explains that the client encrypts a premaster secret (the claimed “encrypt a pre-master key”) using the public key from the server's certificate (the claimed “using the security key of the security engine”), and then sends the result in an encrypted premaster secret message to the server (the claimed “send the encrypted pre-master key to the remote server.”). <i>See TLS-1.2</i> at 58-61 (RSA encrypted premaster secret message); <i>see also TLS-1.2</i> at 5, 50-51, 57-58, 65, 74, 86, 92. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i>

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	provide similar disclosures as <i>TLS-1.2</i>. See <i>SSL-3.0</i> at 28-30, 32-33, 40, 50-51, 55-57; <i>TLS-1.0</i> at 39-40, 43-44, 48, 53-54, 69-71; <i>TLS-1.1</i> at 42, 47-50, 53, 59-60, 75-76. <i>Chou</i> describes a KEK unit 55 of HSM 20 (part of the claimed “security engine”) that “provides an ability to wrap/unwrap keys using AES key wrap specification.” <i>Chou</i> ¶ [0040]; see also <i>id.</i> ¶ [0037] (“Components 54-59 are operable to provide processing operations that pertain to secure key functions, including the wrapping and unwrapping of keys, context data, etc.”), ¶ [0069] (“Secure keys are wrapped/unwrapped and authenticated within the secure zone and do not leave the HSM in an unwrapped state . . .”). A POSITA would have found it obvious when implementing <i>Bugiel</i>’s TruWalletM method to use <i>Chou</i>’s KEK unit 55 to wrap the pre-master key before sending the encrypted pre-master key to the server. <i>Chou</i>’s Figure 3 below illustrates that PKA module 56 and KEK Unit 55 of HSM 20 are communicatively coupled via bus 71. See <i>Chou</i> ¶ [0035] (“Other units 54-69 are also coupled to bus 71.”). Additionally, <i>Chou</i>’s KEK Unit 55 communicates with the non-secure zone through interfaces 42, bus 72 and IPC Msg RAM 61. See <i>Chou</i> ¶¶ [0035]-[0036] (“IPC/Msg RAM module 61 operates as an interface/bridge between bus 71 and bus 72, as well as the only access pathway to exchange data between HSM 20 and the non-secure domain.”). Using these connections, <i>Chou</i>’s KEK unit 55 would load the pre-master key generated in PKA module 56 and the server public key passed through the security interfaces to encrypt the pre-master key using the server public key. In this way, a POSITA would have found it obvious to use <i>Chou</i>’s hardware to encrypt the pre-master key using the server public key as <i>Bugiel</i> describes. See <i>Bugiel</i> at 225, 231.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	FIG. 3 In addition to <i>Chou</i>, as explained at the beginning of this contention, each one of <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i> also teaches security engines of the SoC apparatus in other combinations based on <i>Bugiel</i> are capable of performing encryption. <i>See also supra</i> [1pre]-[1a-1], [1a-3], [1d]-[1e], [1g]. Thus, in this way, like the '838 patent, the grounds alleged here use the security engine (<i>see supra</i> [1a-1]) to implement secure communication using known protocols (IPsec or SSL/TLS protocol, e.g., <i>TLS-1.2</i>) as <i>Bugiel</i> describes, including encrypting the pre-master key with the server's public key according to the SSL/TLS Handshake Protocol.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Third, like the '838 specification (<i>see</i> '838 pat. at 11:6-9), the grounds alleged here include storing in the secure memory, and in an encrypted state (encrypted using the security key), the keys and related keying data generated during the client key exchange. In the alleged combinations, the security engine discussed in [1a-1] implements <i>Bugiel</i>'s teaching to store the pre-master key in the secure storage and in an encrypted state (encrypted using the security key). More specifically, <i>Bugiel</i> explains that “the SSL/TLS session key Sc is generated (which is never disclosed to WalletHelper) and sealed (together with pre-master secret PMS and the hash value hash(CertS)). The resulting session-token STc is stored in a secure storage.” <i>Bugiel</i> at 225; <i>see also id.</i> at Fig. 2, 231. <i>Bugiel</i> further explains that “[s]ealing means protecting an object so that only a certain set of entities can access or use it,” <i>id.</i> at 225-227 & n.8, and that “[w]hen sealed, data is encrypted under a key that is never available outside the TrEE,” <i>id.</i> at 231; <i>see also id.</i> at Fig. 2. As explained for claim feature [1a-3], a POSITA would have found it obvious to use the claimed “security key” discussed in [1a-3], which is used to encrypt other keys. For example, a POSITA would have found it obvious to use <i>Chou</i>'s HSM 20 (part of the claimed “security engine”) to implement this aspect of <i>Bugiel</i>'s method in the secure execution environment based on known protocols for secure communication (e.g., IPsec IKE or SSL/TLS), where KEK 55 would encrypt the PMS (pre-master secret) using <i>Chou</i>'s device unique key/root key stored in OTP 54. <i>See supra</i> [1a-1], [1a-3], [1e]. As <i>Chou</i>'s Figure 3 shows below, PKA module 56 and KEK Unit 55 of the HSM are communicatively coupled by bus 71. <i>See Chou</i> ¶ [0035] (“Other units 54-69 are also coupled to bus 71”). Additionally, “a direct link exists between KEK Unit 55 and OTP so that the key stored in OTP 54 may be loaded into KEK unit 55 without transitioning onto bus 71.” <i>Id.</i> ¶ [0039]. In this way, KEK unit 55 loads the pre-master secret generated by PKA module 56 and the device unique key/root key encoded in OTP 54 to encrypt the pre-master key using the device unique key/root key. A POSITA thus would have found it obvious to use <i>Chou</i>'s security hardware (which provides the secure execution environment <i>Bugiel</i> describes) to encrypt the claimed “pre-master key” (<i>Bugiel</i>'s pre-master secret, PMS) using the claimed “security key” (<i>Chou</i>'s device unique key/root key, which is an encryption key never available outside the TrEE, <i>see Chou</i> ¶ [0025], ¶ [0040]) as <i>Bugiel</i> teaches. <i>See Bugiel</i> at 225 & n.8, 231. HSM 20 also stores the encrypted pre-master key (<i>Bugiel</i>'s encrypted pre-master secret) in <i>Bugiel</i>'s secure storage (<i>see Bugiel</i> at 222-225, 228, 231-234), which could be

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>Chou</i>'s secure memory 22, which “store[s] data that is processed and/or to be processed by security processor 21.” <i>Chou</i> ¶ [0021]; <i>see also supra</i> [1pre]-[1a-3], [1e]-[1f]. FIG. 3 <i>Chou</i>, Fig. 3. As another example, a POSITA would have found it obvious to use <i>Tkacik</i>'s security engine (e.g., secure processing section 30) to implement this aspect of <i>Bugiel</i>'s method in the secure execution environment based on known protocols for secure communication (e.g., IPsec IKE or SSL/TLS), where the secure processing section 30 would encrypt the PMS (pre-master secret) using <i>Tkacik</i>'s master key 160. <i>See Tkacik</i> ¶¶ [0079]-[0080] (generating keys based using IKE, DH, SSL, RSA key gen . . .), ¶¶ [0083]-[0084] (explaining “plaintext master key 160 more directly serves as the BKEK . . . a task 162 moves the plaintext active key 172 generated above in task 144 to data register 76 and encrypts this key 172 using encryption process 70' of SS algorithm 70 and the BKEK”), ¶ [0037],

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	¶ [0055]. <i>See supra</i> [1a-1], [1a-3], [1c]-[1e]. <i>TLS-1.2</i> also explains that the result of RSA key gen and Diffie-Hellman in SSL/TLS protocol is the pre-master secret. <i>See TLS-1.2</i> at 92-93 (explaining “the client and server will generate the same Diffie-Hellman result (i.e., the pre_master_secret”), 58 (“If RSA is being used for key agreement and authentication, the client generates a 48-byte premaster secret”); <i>see also TLS-1.2</i> at 5, 34, 50-51, 53, 56-60, 64-65, 74, 86-87, 91-93; <i>see also SSL-3.0</i> at 28-30, 32-33, 40, 51, 55-57; <i>TLS-1.0</i> at 29, 39-40, 43-45, 47-48, 53, 55, 69-71, 74; <i>TLS-1.1</i> at 31, 42, 47-50, 52-53, 59-60, 75-76; Therefore, a POSITA would have understood <i>Tkacik</i>’s plaintext active key 172, which is generated based by Diffie-Hellman or RSA in the context of TLS/SSL, would also be the <i>TLS-1.2</i> pre-master key in this context.¹⁷ <i>Tkacik</i> further teaches encrypting the plaintext active key 172 by using the master key 160 (the claimed “security key”). <i>See Tkacik</i> ¶¶ [0083]-[0084] (explaining “plaintext master key 160 more directly serves as the BKEK . . . a task 162 moves the plaintext active key 172 generated above in task 144 to data register 76 and encrypts this key 172 using encryption process 70’ of SS algorithm 70 and the BKEK”). The secure processing section 30 also stores the encrypted pre-master key (<i>Bugiel</i>’s encrypted pre-master secret) in <i>Bugiel</i>’s secure storage (<i>see Bugiel</i> at 222-225, 228, 231-234), which could be the secure memory within <i>Tkacik</i>’s secure processing section 30. <i>See Tkacik</i> ¶ [0054]; <i>see also supra</i> [1pre]-[1a-3], [1e]-[1f]. In addition, a POSITA would have found it obvious to use <i>Paaske</i>’s security engine (e.g., TPM) to implement this aspect of <i>Bugiel</i>’s method in the secure execution environment based on known protocols for secure communication (IPsec IKE or SSL/TLS), where the TPM would encrypt the PMS using <i>Paaske</i>’s “keys stored in the OTP memory [, which] may be used to encrypt other keys.” <i>See supra</i> [1a-1], [1a-3], [1c]-[1e]; <i>see also Paaske</i> ¶¶ [0054]-[0056], ¶ [0066], ¶¶ [0112]-[0114], ¶ [0005], ¶ [0007], ¶ [0023], ¶¶ [0028]-[0029], ¶ [0031], ¶ [0034], ¶¶ [0042]-[0044], ¶ [0047], ¶¶ [0054]-[0056], ¶¶ [0059]-[0062], ¶¶ [0101]-[0102], ¶¶ [0110]-[0112], ¶ [0114], ¶ [0124], ¶ [0141], ¶ [0149],

¹⁷ *Tkacik*’s plaintext active key 172 refers to any key used in an applicable cryptographic algorithm. *See, e.g., Tkacik* ¶ [0055] (“The selected algorithm uses plaintext active key 172 stored in active key register 72 to perform the selected cryptographic algorithm on the payload data stored in data register 76.”), ¶ [0080] (“keys may be generated using any of a number of protocols known to those skilled in the art, including IKE, DH, RSA key gen., SSL, random key, and the like”). A POSITA would have understood TLS pre_master_secrets, master_secrets, and key_blocks, for instance, are all examples of a plaintext active key 172 because each is used to perform certain TLS cryptographic algorithms.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	¶¶ [0154]-[0156], ¶¶ [0167]-[0168], ¶ [0179], ¶ [0188], ¶¶ [0190]-[0191], ¶¶ [0217]-[0227], Figs. 1-3. The TPM also stores the encrypted pre-master key (<i>Bugiel</i>'s encrypted pre-master secret) in <i>Bugiel</i>'s secure storage (see <i>Bugiel</i> at 222-225, 228, 231-234), which could be the secure memory located within the TPM. See <i>supra</i> [1pre]-[1a-3], [1e]-[1f]. In addition, a POSITA would have found it obvious to use <i>Tugenberg</i>'s security engine (e.g., host processor 16 and secure memory) to implement this aspect of <i>Bugiel</i>'s method in the secure execution environment based on known protocols for secure communication (IPsec IKE or SSL/TLS, etc.), where host processor 16 and secure memory 20 (together, the claimed "security engine") would encrypt the PMS using <i>Tugenberg</i>'s laser-scribed encryption key 21. See <i>Tugenberg</i> at 5:49-54 (support for SSL/TLS), 4:20-31 ("each secret key having been encrypted with laser-scribed encryption key 21 using secure processing system 14. When needed, communication device decrypts one of the secret keys with encryption logic circuitry 23 using laser-scribed encryption key 21, and stores the decrypted secret key in zeroizable memory 24. The decrypted secret key may be used by host processor 16 in any number of ways, including, for communications with other communication devices"), 4:38-49 ("the secret key may be secret information used to generate a common session key with another communication device. The secret key may also be the common session key used to subsequently communicate secured information"), claim 5(reciting "generating the session key using the secret key and information provided by the destination"); see also <i>supra</i> [1a-1], [1a-3], [1c]-[1e]. The security engine also stores the encrypted pre-master key (<i>Bugiel</i>'s encrypted pre-master secret) in <i>Bugiel</i>'s secure storage (see <i>Bugiel</i> at 222-225, 228, 231-234), which could be <i>Tugenberg</i>'s secure memory. See <i>supra</i> [1pre]-[1a-3], [1e]-[1f]. In addition, a POSITA would have found it obvious to use <i>Aaraj</i>'s security engine (e.g., TPM in the protected execution domain) to implement this aspect of <i>Bugiel</i>'s method in the secure execution environment based on known protocols for secure communication (IPsec IKE or SSL/TLS, etc.), where the security engine would encrypt the PMS using <i>Aaraj</i>'s EK and/or the SRK derived based on EK under Daedalus's apparent view. See <i>supra</i> [1a-1], [1a-3], [1c]-[1e]. The security engine also stores the encrypted pre-master key (<i>Bugiel</i>'s encrypted pre-master secret) in <i>Bugiel</i>'s secure storage

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	(see <i>Bugiel</i> at 222-225, 228, 231-234), which could be <i>Aaraj</i>'s secure memory. See <i>supra</i> [1pre]-[1a-3], [1e]-[1f]. In addition, a POSITA would have found it obvious to use <i>Gibod</i>'s security engine (e.g., the SPU) to implement this aspect of <i>Bugiel</i>'s method in the in the secure execution environment based on known protocols for secure communication (IPsec IKE or SSL/TLS, etc.), where the security engine would encrypt the PMS using <i>Gibod</i>'s Master key. See <i>supra</i> [1a-1], [1a-3], [1c]-[1e]. The security engine also stores the encrypted pre-master key (<i>Bugiel</i>'s encrypted pre-master secret) in <i>Bugiel</i>'s secure storage (see <i>Bugiel</i> at 222-225, 228, 231-234), which could be <i>Gibod</i>'s protected memory. See <i>supra</i> [1pre]-[1a-3], [1e]-[1f]. In addition, a POSITA would have found it obvious to use <i>Mujtaba</i>'s security engine (e.g., wireless communication processor 101) to implement this aspect of <i>Bugiel</i>'s method in the in the secure execution environment based on known protocols for secure communication (IPsec IKE or SSL/TLS, etc.), where the security engine would encrypt the PMS using <i>Mujtaba</i>'s storage key and/or UID. See <i>supra</i> [1a-1], [1a-3], [1c]-[1e]. The security engine also stores the encrypted pre-master key (<i>Bugiel</i>'s encrypted pre-master secret) in <i>Bugiel</i>'s secure storage (see <i>Bugiel</i> at 222-225, 228, 231-234), which could be <i>Mujtaba</i>'s secure memory. See <i>supra</i> [1pre]-[1a-3], [1e]-[1f]. In addition, a POSITA would have found it obvious to use <i>Buer754</i>'s security engine (e.g., security module, cryptographic accelerators) to implement this aspect of <i>Bugiel</i>'s method in the in the secure execution environment based on known protocols for secure communication (e.g., IPsec IKE or SSL/TLS, etc.), where the security engine would encrypt the PMS using <i>Buer754</i>'s KEK. See <i>supra</i> [1a-1], [1a-3], [1c]-[1e]. The security engine also stores the encrypted pre-master key (<i>Bugiel</i>'s encrypted pre-master secret) in <i>Bugiel</i>'s secure storage (see <i>Bugiel</i> at 222-225, 228, 231-234), which could be <i>Buer754</i>'s secure memory. See <i>supra</i> [1pre]-[1a-3], [1e]-[1f]. In sum, the grounds alleged here provide the same disclosures as the specification of the '838 patent as it relates to claim 3. To the extent that Daedalus argues claim 3 is supported and enabled by the specification of the '838 patent, the grounds alleged here satisfy claim 3: "wherein the security engine

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	is further to generate a pre-master key, encrypt the pre-master key using the security key of the security engine, and send the encrypted pre-master key to the remote server.” Exemplary disclosures of the prior art include the following: <i>Bugiel</i> discloses the following at 225-227 (4.1 Establishing SSL/TLS connection and managing two logical SSL/TLS subchannels): Figure 2 illustrates the SSL handshake protocol. The client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters. In the following steps the server's certificate Certs is sent and verified, the SSL/TLS session key Sc is generated (which is never disclosed to WalletHelper) and sealed¹¹ (together with pre-master secret PMS and the hash value hash(Certs)). The resulting session-token STc is stored in a secure storage and can be loaded and unsealed later using the session identifier SID. [Note ⁸: Sealing means protecting an object so that only a certain set of entities can access or use it.] ... All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16]. <i>See also Bugiel</i> at 225-231, 233. <i>See also Paaske</i> ¶ [0005], ¶ [0007], ¶ [0023], ¶¶ [0028]-[0029], ¶ [0031], ¶ [0034], ¶¶ [0042]-[0044], ¶¶ [0047]-[0048], ¶¶ [0054]-[0056], ¶¶ [0059]-[0062], ¶ [0066], ¶¶ [0101]-[0102], ¶ [0106], ¶¶ [0110]-[0114], ¶ [0124], ¶ [0141], ¶ [0149], ¶¶ [0154]-[0156], ¶¶ [0167]-[0169], ¶ [0179], ¶ [0188], ¶¶ [0190]-[0191], ¶ [0194], ¶¶ [0205]-[0206], ¶¶ [0208]-[0210], ¶¶ [0217]-[0227], Figs. 1-3.

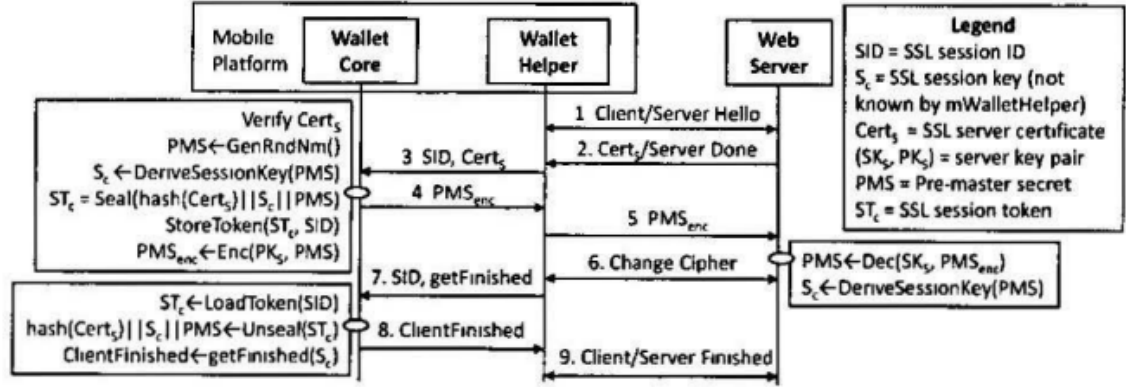
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also Chou</i> ¶ [0021], ¶ [0025], ¶ [0035], ¶¶ [0039]-[0040], ¶¶ [0043]-[0055], Figs. 1 and 3. <i>See also Tkacik</i> Abstract, ¶ [0007], ¶ [0037], ¶ [0055], ¶¶ [0073]-[0074], ¶¶ [0079]-[0080], ¶¶ [0083]-[0084]. <i>See also Tugenberg</i> at 4:20-31, 4:38-49, 5:47-54, claim 5, Figs. 1-2. <i>See also Mujtaba</i> ¶ [0006], ¶¶ [0052]-[0053], ¶¶ [0055]-[0057]. <i>See also Buer754</i> at 1:27-41, 1:61-2:8, 5:24-29, 6:13-31, 9:16-21, 13:14-16:35. <i>See also Gibod</i> ¶ [0037], ¶ [0071]. <i>See also Aaraj</i> at 8:1-8:9, 8:11-8:16, 8:19-8:23, 8:26-8:28, 8:30-8:31. <i>See also Weise</i> ¶ [0010], ¶ [0023], ¶¶ [0033]-[0035], Figs. 3 and 5, claims 9 and 17. <i>See also TLS-1.2</i> at 5, 33-37, 50-53, 56-61, 64-65, 74-75, 86-87, 91-93. <i>See also SSL-3.0</i> at 28-30, 32-33, 40, 50-51, 55-57. <i>See also TLS-1.0</i> at 29, 39-40, 43-45, 47-48, 53-55, 69-71, 74. <i>See also TLS-1.1</i> at 31, 42, 47-50, 52-53, 59-60, 75-76. <i>See supra</i> claim 1.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
[Claim 4]	
[4] The system-on-a-chip apparatus of claim 3 wherein the security engine is further to wrap the pre-master key using a server public key and store the wrapped pre-master key in the secure memory.	The grounds alleged at the beginning of this chart disclose or suggest “the security engine is further to wrap the pre-master key using a server public key and store the wrapped pre-master key in the secure memory.” As an initial matter, claim 4 depends from claim 3, which fails the written description requirement. <i>See supra</i> [3]. To the extent that Daedalus argues the specification of the '838 patent supports claim 3, the grounds alleged here disclose or suggest claim 4. First, the grounds alleged here includes encrypting the pre-master key using a server public key. As Figure 2 indicates, <i>Bugiel</i> teaches encrypting the pre-master key PMS (“$PMS_{enc} \leftarrow (PK_s, PMS)$,” where PK_s is the server public key) and sending the encrypted pre-master key PMS_{enc} to a web server. <i>See Bugiel</i> at 225-226. Additionally, each SSL/TLS protocol also explains that the client wraps the pre-master key using a server public key. For example, <i>TLS-1.2</i> explains that the client encrypts the premaster secret (the claimed “pre-master key”) using the public key from the server’s certificate (the claimed “server public key”). <i>See TLS-1.2</i> at 58-61 (RSA encrypted premaster secret message); <i>see also TLS-1.2</i> at 5, 50-51, 57-58, 65, 74, 86, 91-92. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provide similar disclosures as <i>TLS-1.2</i>. <i>See SSL-3.0</i> at 28-30, 32-33, 40, 50-51, 55-57; <i>TLS-1.0</i> at 39-40, 43-44, 48, 53-54, 69-71; <i>TLS-1.1</i> at 42, 47-50, 53, 59-60, 75-76. <i>See supra</i> claim 3.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	 Fig. 2. SSL/TLS handshake The alleged combination uses the security engine discussed in [1a-1] (<i>see supra</i> [1a-1]) to implement <i>Bugiel</i>'s communications method in the secure execution environment based on known protocols (IPsec IKE or SSL/TLS, etc.), to establish a secure communication. For example, <i>Chou</i>'s HSM 20 (part of the "security engine") implements <i>Bugiel</i>'s teaching to encrypt a pre-master secret with a server public key. In particular, <i>Chou</i>'s KEK unit 55 (part of the claimed "security engine") "provides an ability to wrap/unwrap keys using AES key wrap specification." <i>Chou</i> ¶ [0040]; <i>see also id.</i> ¶ [0037] ("Components 54-59 are operable to provide processing operations that pertain to secure key functions, including the wrapping and unwrapping of keys, context data, etc."), ¶ [0069] ("Secure keys are wrapped/unwrapped and authenticated within the secure zone and do not leave the HSM in an unwrapped state . . ."). As <i>Chou</i>'s Figure 3 shows below, PKA module 56 and KEK Unit 55 are communicatively coupled via bus 71. <i>See Chou</i> ¶ [0035] ("Other units 54-69 are also coupled to bus 71"). Additionally, KEK Unit 55 communicates with the non-secure zone through interfaces 42, bus 72, and IPC Msg RAM 61. <i>See Chou</i> ¶¶ [0035]-[0036] ("IPC/Msg RAM module 61 operates as an interface/bridge between bus 71 and bus 72, as well as the only access pathway to exchange data between HSM 20 and the non-secure domain."). Accordingly, KEK unit 55 would load the pre-master key generated by PKA module 56 and the server public key passed through interfaces

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	42 to encrypt the pre-master key (<i>Bugiel</i>'s pre-master secret PMS) using the server public key (<i>Bugiel</i>'s PKs). A POSITA thus would have found it obvious to use <i>Chou</i>'s security hardware to encrypt the pre-master key using the server public key in the manner <i>Bugiel</i> describes. See <i>Bugiel</i> at 225-227 & n.8, 231. FIG. 3 In addition to <i>Chou</i>, as explained at the beginning of this contention, each one of <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i> also teaches their security engines are capable of performing encryption and related cryptographic functions.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Thus, in this way, like the '838 patent, the grounds alleged here uses the security engine (<i>see supra</i> [1a-1]) to implement in the secure execution environment based on known protocols for secure communication (IPsec or SSL/TLS protocol, e.g., <i>TLS-1.2</i>) as <i>Bugiel</i> describes, including e.g., encrypting the pre-master key with the server's public key according to the SSL/TLS Handshake Protocol. Second, the grounds alleged here includes storing the wrapped pre-master key in secure memory, as claimed. As discussed in [1a-2] and [1f], each of <i>Bugiel</i>, <i>Chou</i>, <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i> teaches the claimed secure memory. For example, one of the assumptions underlying <i>Bugiel</i>'s TruWalletM method is "availability of a secure hardware which provides a Trusted Execution Environment (TrEE) with the following feature . . . (ii) secure storage." <i>Bugiel</i> at 222. <i>Bugiel</i> further explains: "SecureStorage is a database where passwords and other security sensitive data are stored persistently and securely." <i>Bugiel</i> at 224. As <i>Bugiel</i>'s Figure 1 shows, such secure storage is only accessible by the WalletCore in the secure execution environment. And, according to <i>Bugiel</i>, the SecureStorage is isolated "from other, unauthorized (and potentially malicious) scripts." <i>Id.</i> at 231; <i>see also Bugiel</i> at 222-225, 228, 231-234. As another example, <i>Chou</i>'s HSM 20 (part of the claimed "security engine") uses secure memory 22 (part of the claimed "secure memory") to store data that is processed and/or to be processed by security processor. <i>See Chou</i> ¶ [0021]; <i>see also</i> [1a-2] (discussing each reference's disclosure of secure memory), [1f] (discussing storing in secure memory in each reference combinations). Using such hardware discussed in [1pre]-[1a-3], [1e]-[1f], a POSITA would have found it obvious to implement the teaching of <i>Bugiel</i> and/or known protocols for secure communication (IPsec or SSL/TLS, e.g., <i>TLS-1.2</i>) to derive, encrypt, store and send the encrypted pre-master key and derive session key from pre-master key in two ways. First, <i>Bugiel</i> teaches encrypting the pre-master key (pre-master secret PMS) using the server public key (PK_s) and sending the encrypted pre-master key (PMS_{enc}) to a web server. <i>See Bugiel</i> at Fig. 2.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	In addition, each SSL/TLS protocol (e.g., <i>TLS-1.2</i>) teaches storing the pre-master key to compute a master secret using the pre-master key. <i>See TLS-1.2</i> at 64 (explaining the pre-master secret (the claimed “pre-master key”) should be deleted from memory once the master secret has been computed), 93 (“To prevent the pre_master_secret from staying in memory any longer than necessary, it should be converted into the master_secret as soon as possible.”); <i>see also SSL-3.0</i> at 32, 57; <i>TLS-1.0</i> at 47, 71; <i>TLS-1.1</i> at 52, 76. Each SSL/TLS protocol also teaches sending the encrypted pre-master secret to the server. In TLS version 1.2, after wrapping the pre-master secret, the client sends the result in an encrypted premaster secret message (the claimed “wrapped pre-master key”) to the server. <i>See TLS-1.2</i> at 58-61 (RSA encrypted premaster secret message); <i>see also TLS-1.2</i> at 5, 50-51, 57-58, 65, 74, 86, 91-92. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provide similar disclosures as <i>TLS-1.2</i>. <i>See SSL-3.0</i> at 28-30, 32-33, 40, 50-51, 55-57; <i>TLS-1.0</i> at 39-40, 43-44, 48, 53-54, 69-71; <i>TLS-1.1</i> at 42, 47-50, 53, 59-60, 75-76. <i>See supra</i> claim 3. The encrypted pre-master secret is thus data that is processed and to be processed according to <i>Bugiel</i> and in the secure execution environment based on known protocols for secure communication (e.g., IPsec or SSL/TLS, etc.). Therefore, the encrypted pre-master secret is an intermediate result of the TLS handshake. <i>See TLS-1.2</i> at 91 (“The general goal of the key exchange process is to create a pre_master_secret known to the communicating parties and not to attackers. The pre_master_secret will be used to generate the master_secret . . . required to generate the . . . encryption keys . . .”), 58-61 (describing an RSA encrypted premaster secret message). As discussed in [1a-2], [1f] and [3], such intermediate results are stored within the secure execution environment (e.g., the secure memory). A POSITA thus would have found it obvious to store the encrypted pre-master key in the secure memory discussed in [1a-2]. SSL/TLS protocols also teach that secret keys, such as the pre-master key, should be kept securely. For example, <i>TLS-1.2</i> emphasizes the importance of keeping the pre-master key secure and describes the pre-master key as an essential secure parameter for computing the master secret and encryption keys. <i>See TLS-1.2</i> at 92 (“Eavesdroppers who do not know the private values should not be able to find the Diffie-Hellman result (i.e., the pre_master_secret”), 96 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”) (emphases added); <i>see also id.</i> at 1, 4, 34, 59, 91-97. <i>SSL-3.0</i>, <i>TLS-1.0</i> and <i>TLS-1.1</i> also provide similar disclosures. <i>See SSL-3.0</i> at 59 (“For

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	SSL to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 9, 55-58, 17; <i>TLS-1.0</i> at 73 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 13, 29-30, 44-45, 69, 71-72; <i>TLS-1.1</i> at 80 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 14-15, 31, 48-49, 74-75, 77-80. <i>Bugiel</i> also teaches that its method relies on “the TrEE provided by secure hardware is tamper-protected.” <i>Bugiel</i> at 222, 233 (“credentials associated with the SSL/TLS channel (such as pre-master secret key and a session key) are protected by the TrEE”); <i>see also Bugiel</i> at 220-226, 228-235. <i>Chou</i> further explains: “Security processor 21 provides the various processing within HSM 20, including key authentication and the encrypting and/or decrypting operation to encrypt and/or decrypt data. . . . Secure memory 22 is utilized to store data that is processed and/or to be processed by security processor 21.” <i>Chou</i> ¶ [0021]. A POSITA thus would have found it obvious to store the encrypted pre-master key in <i>Chou</i>’s secure memory 22 because, when implementing the method of <i>Bugiel</i> and in the secure execution environment based on known protocols for secure communication (IPsec or SSL/TLS, etc.), the encrypted pre-master key is data processed by the security processor 21 (such as KEK unit 55). In addition to <i>Chou</i>, a POSITA would have found it obvious to store the encrypted pre-master key in the secure memory discussed in [1a-2] as it’s being processed in the corresponding security engine. <i>See supra</i> [1a-1]-[1a-2], [1f] (discussing each reference’s disclosure of security engine and secure memory). Second, under Daedalus’s apparent view of the claims, <i>Bugiel</i> and each known security protocol (e.g., SSL/TLS protocol, etc.) teach storing the encrypted pre-master key when the encrypted pre-master secret is sent from client device to a remote server. <i>See Infringement Contentions</i>, Ex. F-1 at 21-22; <i>supra</i> [1f].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

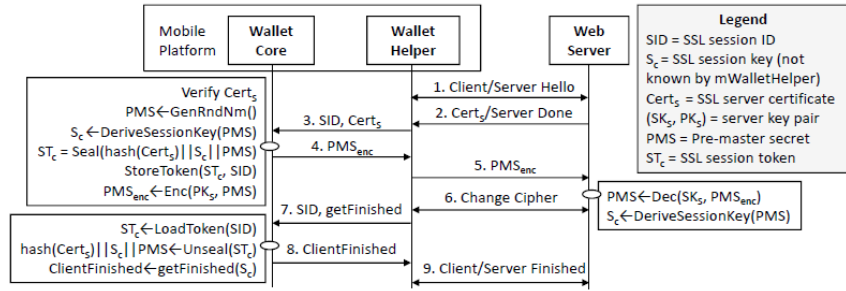
The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Exemplary disclosures of the prior art include the following: <i>Bugiel</i> discloses the following at 225-227 (4.1 Establishing SSL/TLS connection and managing two logical SSL/TLS subchannels): Figure 2 illustrates the SSL handshake protocol. The client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters. In the following steps the server's certificate $Cert_s$ is sent and verified, the SSL/TLS session key S_c is generated (which is never disclosed to WalletHelper) and sealed¹¹ (together with pre-master secret PMS and the hash value $\text{hash}(Cert_s)$). The resulting session-token ST_c is stored in a secure storage and can be loaded and unsealed later using the session identifier SID. [Note ⁸: Sealing means protecting an object so that only a certain set of entities can access or use it.] ... All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16].  <pre> sequenceDiagram participant MP as Mobile Platform participant WC as Wallet Core participant WH as Wallet Helper participant WS as Web Server Note over MP: Verify Cert_s PMS ← GenRndNm() Note over MP: S_c ← DeriveSessionKey(PMS) Note over MP: ST_c = Seal(hash(Cert_s) S_c PMS) Note over MP: StoreToken(ST_c, SID) Note over MP: PMS_enc ← Enc(PK_s, PMS) WH->>WS: 1. Client/Server Hello WS->>WH: 2. Cert_s/Server Done WH->>WC: 3. SID, Cert_s WC->>WH: 4. PMS_enc WH->>WS: 5. PMS_enc WS->>WH: 6. Change Cipher WH->>WC: 7. SID, getFinished Note over WS: PMS ← Dec(SK_s, PMS_enc) Note over WS: S_c ← DeriveSessionKey(PMS) MP->>WC: ST_c ← LoadToken(SID) Note over MP: hash(Cert_s) S_c PMS ← Unseal(ST_c) Note over MP: ClientFinished ← getFinished(S_c) MP->>WH: 8. ClientFinished WH->>WS: 9. Client/Server Finished </pre> Legend SID = SSL session ID S_c = SSL session key (not known by mWalletHelper) $Cert_s$ = SSL server certificate (SK_s, PK_s) = server key pair PMS = Pre-master secret ST_c = SSL session token

Fig. 2. SSL/TLS handshake

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also Bugiel</i> at 220-236. <i>See also TLS-1.2</i> at 1, 4-5, 34, 50-51, 57-61, 64-65, 74-75, 86, 91-97. <i>See also SSL-3.0</i> at 1, 9, 17, 28-30, 32-33, 40, 50-51, 55-59. <i>See also TLS-1.0</i> at 1, 4, 13, 29-30, 39-40, 43-45, 47-48, 53-54, 69-74. <i>See also TLS-1.1</i> at 1, 4, 14-15, 31, 42, 47-50, 52-53, 59-60, 74-80. <i>See supra</i> claims 1 and 3.
Claim 5	
[5a] The system-on-a-chip apparatus of claim 3, wherein the security engine is further to: generate the session key as a function of the pre-master key; and	The grounds alleged at the beginning of this chart disclose or suggest “the security engine is further to: generate the session key as a function of the pre-master key.” Claim 5 depends from claim 3, which fails the written description requirement. <i>See supra</i> [3]. Therefore, claim 5 also fails the enablement and written description requirements due to its dependency on claim 3. To the extent that Daedalus argues the specification supports claim 3, the grounds alleged here disclose or suggest “the security engine is further to: generate the session key as a function of the pre-master key.” <i>Bugiel</i> teaches generating the session key Sc (the claimed “session key”) as a function of the pre-master secret PMS (the claimed “pre-master key”) by explaining: “Sc←DeriveSessionKey(PMS)” (the claimed “function of the pre-master key. <i>See Bugiel</i> at Fig. 2; <i>see also Bugiel</i> at 225-226. The grounds alleged here uses the security engine discussed in [1a-1] to implement <i>Bugiel</i>’s communications method in the secure execution environment based on known protocols (IPsec IKE or

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	SSL/TLS, etc.) to establish a secure communication, including the teaching to generate the session key as a function of the pre-master key. For example, <i>Chou</i> explains that PKA module 56 (part of the claimed “security engine”) performs cryptographic key exchange algorithms, such as Diffie-Hellman shared key computation and RSA key generation, to generate secure keys. <i>See supra</i> [1c], [1d], [3]; <i>Chou</i> ¶¶ [0043]-[0053], Fig. 3 (showing the PKA module 56 generates secure keys and transports them to key cache 65 so that SPU-M 39 can perform data encryption/decryption). <i>See also supra</i> [1a-1], [1a-3], [1c]-[1e], [3] (describing each references’ support for cryptographic operations such as key exchange and/or key encryption using security engine). FIG. 3 More specifically, each SSL/TLS protocol teaches generating a session key S_c as a function of the pre-master key in two ways.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	First, the TLS/SSL master secret may be the claimed “symmetric session key” under Daedalus’s view. <i>See</i> [1d]. Under such a mapping, <i>TLS-1.2</i> teaches to generate a master secret (the claimed “session key”) as a PRF function (the claimed “function”) of pre-master secret (the claimed “pre-master key”). <i>See TLS-1.2</i> at 64 (Computing the master secret), 14-15 (describing PRF); <i>see also TLS-1.2</i> at 4-6, 14-19, 25-27, 33-37, 58, 63-65, 77-82, 91, 94-95. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provide similar disclosures. <i>See SSL-3.0</i> at 32 (computing master secret from pre_master_secret based on MD5), 58 (MD5); <i>see also SSL-3.0</i> at 11, 29-34, 40, 45, 54-59; <i>TLS-1.0</i> at 47 (computing the master secret); <i>see also TLS-1.0</i> at 11-15, 21-23, 29, 45-48, 56, 59, 68-72; <i>see also TLS-1.1</i> at 12-16, 24-25, 31, 48, 51-53, 64, 66, 74-78. 8.1. Computing the Master Secret For all key exchange methods, the same algorithm is used to convert the pre_master_secret into the master_secret. The pre_master_secret should be deleted from memory once the master_secret has been computed. <pre style="margin-left: 120px;"> master_secret = PRF(pre_master_secret, "master secret", ClientHello.random + ServerHello.random) [0..47]; </pre> Second, and in the alternative, <i>TLS-1.2</i> also teaches generating a key_block (the claimed “session key”) for data encryption, based on the master secret for each session. <i>Id.</i> at 25-26 (Key Calculation); <i>see also TLS-1.2</i> at 4, 6, 12-14, 16-18, 22-28, 33, 35, 40, 58, 63, 65, 69, 75-84, 86, 91, 94-100; <i>see supra</i> claim feature [1d]. <pre style="margin-left: 120px;"> key_block = PRF(SecurityParameters.master_secret, "key expansion", SecurityParameters.server_random + SecurityParameters.client_random); </pre> Because the master secret is a PRF function of pre-master secret, the key_block, which is a function of master secret, is thus also a PRF function of pre-master secret.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	The key_block is the claimed “session key” because it is used to encrypt messages sent to the remote server and to decrypt messages received from the remote server during a secure communication session. More specifically, the key_block is a key with multiple partitions, including a “client_write_key” partition and a “server_write_key” partition. <i>See TLS-1.2</i> at 25-26 (Key Calculation). The client uses the client_write_key partition to encrypt data transmitted from client to server, and the client uses the server_write_key partition to decrypt data received from server. <i>See id.</i> at 79-81 (defining master secret, client_write_key partition, server_write_key partition). <i>TLS-1.2</i> further explains that “TLS secures transmitted application data via the use of symmetric encryption,” indicating the encryption keys are symmetric session keys. <i>Id.</i> at 95. The key_block generated based on master secret thus also reads on the claimed “symmetric session keys.” <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> also provide similar disclosures as <i>TLS-1.2</i>. <i>See SSL-3.0</i> at 4, 9-11, 14-17, 19, 26, 29-37, 42-46, 48, 50-51, 55, 58-59, 61; <i>see also TLS-1.0</i> at 3-4, 10, 12-16, 18-19, 21-23, 26, 30, 35, 46, 49, 55-60, 62, 64, 69, 72-73, 76; <i>see also TLS-1.1</i> at 4, 6, 11, 13, 15-17, 20, 24-26, 28, 33, 37, 55, 63-70, 75, 78-81. In addition, each IKE protocol (<i>RFC-5996</i> or <i>RFC-4306</i>) also teaches to generate the session key as a function of the pre-master key. More specifically, <i>RFC-5996</i> discloses generating g^{ir} in the initial exchange or g^{ir} (new) (either could be the claimed premaster key) in the Child SA exchange. <i>See RFC-5996</i> at 47 (“g^{ir} is the shared secret from the ephemeral Diffie-Hellman exchange.”), 52 (“g^{ir} (new) is the shared secret from the ephemeral Diffie-Hellman exchange of this CREATE_CHILD_SA exchange”). <i>RFC-5996</i> then discloses generating a series of keys, such as SKEYSEED, Sk_d, Sk_ai, Sk_ar, Sk_ei, Sk_er, Sk_pi, Sk_pr, KEYMAT based on g^{ir} or g^{ir} (new). <i>See RFC-5996</i> at 46-47, 52-53, 11, 13. Any of these keys could be the claimed session key under Daedalus’ apparent view.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on Bugiel

The '838 Patent	Exemplary Disclosures Relevant to Bugiel and Related Combinations
	KEYSEED and its derivatives are computed as follows: $\text{KEYSEED} = \text{prf}(\text{Ni} \parallel \text{Nr}, g^{\text{ir}})$ $\{\text{SK}_d \parallel \text{SK}_{ai} \parallel \text{SK}_{ar} \parallel \text{SK}_{ei} \parallel \text{SK}_{er} \parallel \text{SK}_{pi} \parallel \text{SK}_{pr}\} \\ = \text{prf}^+ (\text{KEYSEED}, \text{Ni} \parallel \text{Nr} \parallel \text{SPI}_i \parallel \text{SPI}_r)$ RFC-5996 at 47. RFC-4306 provides similar disclosures. See RFC-4306 at 8, 10, 28-29, 33-34. Exemplary disclosures of the prior art include the following: Bugiel discloses the following at Fig. 2: Fig. 2. SSL/TLS handshake See also Bugiel at 225-226, 235-236. See also Chou ¶¶ [0043]-[0053], Fig. 3

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also TLS-1.2</i> at 4-6, 12-19, 22-28, 33-37, 40, 58-61, 63-65, 69, 75-84, 91, 94-100. <i>See also SSL-3.0</i> at 4, 9-11, 14-17, 19, 26, 29-37, 40, 42-46, 48, 50-51, 54-59, 61. <i>See also TLS-1.0</i> at 3-4, 10-16, 18-19, 21-23, 26, 29-30, 35, 45-49, 55-60, 62, 64, 68-73, 76. <i>See also TLS-1.1</i> at 4, 6, 11-17, 20, 24-26, 28, 31, 33, 37, 48, 51-53, 55, 63-70, 74-81. <i>See also RFC-5996</i> at 11, 13, 46-47, 52-53. <i>See also RFC-4306</i> at 8, 10, 28-29, 33-34. <i>See supra</i> claims 1, 3-4.
[5b] calculate a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server.	The grounds alleged at the beginning of this chart disclose or suggest the security engine to “calculate a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server.” <i>Bugiel</i> teaches performing hash algorithm. <i>See Bugiel</i> at 225 (“the hash value hash(Certs)”); <i>see also Bugiel</i> at 226-230, 232. <i>Bugiel</i> explains: “All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16].” <i>Bugiel</i>, at 225-226. <i>Bugiel</i> for example invokes reference [16], which is <i>TLS-1.2</i>, for how to implement the cryptographic key exchange messages with a remote Web Server, including calculating a hash function. <i>See id.</i> The combinations alleged here use the security engine discussed in [1a-1] (<i>see supra</i> [1a-1] on disclosures of security engine from each reference) to implement <i>Bugiel</i>’s communications method, including to implement in the secure execution environment based on known protocols (IPsec IKE or SSL/TLS Handshake Protocol messages, etc.), to establish a secure communication.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	For example, <i>Chou</i>'s HSM 20 (part of the claimed "security engine") implement known protocols for secure communication (e.g., IPsec IKE or SSL/TLS Handshake Protocol messages, etc.) <i>Bugiel</i> teaches, including e.g., perform cryptographic key exchange to generate keys using the TLS Handshake protocol. Each of the security engine discussed in [1a-1] can be used for such implementation. For example, <i>Bugiel</i> discloses implementing its method in M-Shield. <i>See Bugiel</i> at 230, 222-224, 234-235, FIG. 1. <i>Chou</i> is also designed to calculate a hash function. <i>See Chou</i> ¶¶ [0043]-[0053] (explaining that PKA module 56 (part of the claimed "security engine") may perform algorithms like "1024-bit DSA signature with SHA-1 hash or 2048-bit DSA signature with SHA-256 hash"), ¶¶ [0055]-[0062] (SHA1-HMAC, SHA2-HMAC, MD5-HMAC, etc.). In addition to <i>Chou</i>, other references discussed in [1a-1] also disclose a security engine for performing related cryptographic function, such as hashing. <i>See, e.g., Paaske</i> ¶¶ [0054]-[0055], ¶ [0062], ¶ [0105], ¶ [0124], ¶ [0127], ¶ [0131], ¶ [0141], ¶ [0149], ¶ [0153], ¶¶ [0156]-[0157], ¶ [0175], ¶ [0182], ¶ [0191], ¶ [0200], ¶ [0214].; <i>Tkacik</i> ¶ [0052] ("encryption engine 66 may also perform other cryptographic algorithms, such as hashing algorithm"), ¶ [0059], ¶ [0067], ¶¶ [0069]-[0070], ¶ [0074], ¶ [0077], ¶¶ [0080]-[0081], ¶ [0083]; <i>Tugenberg</i> at 5:47-54; <i>Aaraj</i> at 8:7 (HMAC and SHA-1 hashing functions), 8:5, 8:9, 8:11-8:15; <i>Gibod</i> ¶¶ [0033]-[0037], ¶ [0071], Fig. 1; <i>Mujtaba</i> at ¶ [0054]; <i>Buer754</i> at 15:4-11 (HMAC-SHA-1), 1:27-41 (IPsec, IKE, SSL), 1:61-2:8, 5:24-29, 6:13-31, 8:63-9:4, 9:16-21, 13:21-37, 13:45-65, 14:11-34, 14:43-55, 15:4-11, 15:17-16:19, 16:26-35, Fig. 14; <i>Weise</i> ¶ [0010] (SSL, IPsec), ¶ [0034] (secure hash algorithm), ¶ [0023], ¶¶ [0034]-[0035], claims 9 and 17; <i>see also supra</i> [1a-1]. During the TLS Handshake protocol as implemented using the alleged combination, a hash function must be calculated as a function of the random nonce generated by the security engine and a nonce received from the remote server. More specifically, each SSL/TLS protocol teaches calculating a PRF or MD5/SHA function (the claimed "hash function") as a function of ClientHello.random (the claimed "random nonce generated by the security engine") and ServerHello.random (the claimed "nonce received from the remote server"). <i>See TLS-1.2</i> at 14-15 (PRF function); <i>see also TLS-1.1</i> at 12-14 (PRF function); <i>TLS-1.0</i> at 11-13 (PRF function); <i>SSL-3.0</i> at 26 (md5_hash and sha_hash), 58 (MD5 and SHA). Here,

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	ClientHello.random is random nonce generated by the client, and ServerHello.random is the random nonce received from the server. <i>See TLS-1.2</i> at 34 (Handshake Protocol Overview); <i>see also SSL-3.0</i> at 18; <i>TLS-1.0</i> at 30; <i>TLS-1.1</i> at 32; <i>see also</i> claim features [1b] and [2]. Each SSL/TLS protocol discloses the claimed feature in at least four ways: First, each SSL/TLS protocol discloses or suggests calculating the master secret involves the claimed “calculate a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server.” <i>TLS-1.2</i> teaches the following: “master_secret = PRF (pre_master_secret, “master secret”, ClientHello.random + ServerHello.random).” <i>TLS-1.2</i> at 64 (Computing the Master Secret). <i>TLS-1.2</i> further teaches: “TLS’s PRF is created by applying P-hash to the secret as: PRF(secret, label, seed) = P_<hash>(S1, label + seed).” <i>Id.</i> at 14-15 (HMAC and the Pseudorandom Function); <i>see also TLS-1.2</i> at 5-6, 16-18, 25-27, 40, 63, 66, 77, 80-81, 84, 92, 94, 97. As these disclosures show, creating the master_secret in TLS version 1.2 involves calculating a hash function (PRF function) as a function of the random nonce generated by the security engine and a nonce received from the remote server (the ClientHello.random and ServerHello.random, respectively). <i>TLS-1.1</i> and <i>TLS-1.0</i> provide similar disclosures as <i>TLS-1.2</i>. <i>See TLS-1.1</i> at 12-14, 16, 24, 44, 51-53, 59, 66-67, 76, 78, 81-82; <i>TLS-1.0</i> at 11-13, 15, 21-23, 41, 46-47, 52-53, 59-60, 71-72, 76-77. In <i>SSL-3.0</i>, MD5 and SHA hash functions of random nonces are performed instead of PRF function. <i>See SSL-3.0</i> at 32 (“master_secret = MD5(pre_master_secret + SHA('A' + pre_master_secret + ClientHello.random + ServerHello.random)) + MD5(pre_master_secret + SHA('BB' + pre_master_secret + ClientHello.random + ServerHello.random)) + MD5(pre_master_secret + SHA('CCC' + pre_master_secret + ClientHello.random + ServerHello.random)))”); <i>see also SSL-3.0</i> at 11, 26-27, 30-35, 39, 41, 45-46, 58, 61-62.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Second, each SSL/TLS protocol discloses or suggests calculating the encryption key_block involves the claimed “calculate a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server.” <i>TLS-1.2</i> teaches the following: “key_block = PRF(SecurityParameters.master_secret, ‘key expansion’, SecurityParameters.server_random + SecurityParameters.client_random.” <i>Id.</i> at 26, 94 (“The master secret is hashed with the ClientHello.random and ServerHello.random to produce unique data encryption keys . . .”); <i>see also TLS-1.2</i> at 5-6, 16-18, 25-27, 40, 63, 66, 77, 80-81, 84, 92, 94, 97. Therefore, creating encryption key_block also involves calculating a hash function (the PRF function) as a function of the random nonce generated by the security engine and a nonce received from the remote server (the ClientHello.random and ServerHello.random, respectively). <i>TLS-1.1</i> and <i>TLS-1.0</i> provide similar disclosures as <i>TLS-1.2</i>. <i>See TLS-1.1</i> at 12-14, 16, 24, 44, 51-53, 59, 66-67, 76, 78, 81-82; <i>TLS-1.0</i> at 11-13, 15, 21-23, 41, 46-47, 52-53, 59-60, 71-72, 76-77. In <i>SSL-3.0</i>, MD5 and SHA hash functions of random nonces are performed instead of PRF function. <i>See SSL-3.0</i> at 34 (“key_block = MD5(master_secret + SHA('A' + master_secret + ServerHello.random + ClientHello.random)) + MD5(master_secret + SHA('BB' + master_secret + ServerHello.random + ClientHello.random)) + MD5(master_secret + SHA('CCC' + master_secret + ServerHello.random + ClientHello.random)) + [...]”), 34 (“final_client_write_key = MD5(client_write_key + ClientHello.random + ServerHello.random); final_server_write_key = MD5(server_write_key + ServerHello.random + ClientHello.random);”); <i>see also SSL-3.0</i> at 11, 26-27, 30-35, 39, 41, 45-46, 58, 61-62. Third, each SSL/TLS protocol discloses or suggests generating a Finished message involves the claimed “calculate a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server.” Each SSL/TLS protocol teaches that the client will generate a Finished message that includes a hash of all the handshake messages. For instance, <i>TLS</i> version 1.2 calculates the following: “PRF(master_secret, finished_label, Hash(handshake_message) [0 . . verify_data_length-1],” where handshake_message includes all the handshake messages, including ClientHello and ServerHello

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	messages. <i>See TLS-1.2</i> at 63-64 (Finished); <i>see also TLS-1.2</i> at 5-6, 16-18, 25-27, 40, 63, 66, 77, 80-81, 84, 92, 94, 97. Because ClientHello includes ClientHello.random, and because ServerHello includes ServerHello.random, creating the Finished message also involves calculating a hash function (the PRF function) as a function of the random nonce generated by the security engine and a nonce received from the remote server (the ClientHello.random and ServerHello.random, respectively). <i>TLS-1.1</i> and <i>TLS-1.0</i> provide similar disclosures as <i>TLS-1.2</i>. <i>See TLS-1.1</i> at 51 (“PRF(master_secret, finished_label, MD5(handshake_messages) + SHA-1(handshake_message)) [0..11]”); <i>see also TLS-1.1</i> at 12-14, 16, 24, 44, 51-53, 59, 66-67, 76, 78, 81-82; <i>TLS-1.0</i> at 46 (“PRF(master_secret, finished_label, MD5(handshake_messages) + SHA-1(handshake_message)) [0..11]”); <i>see also TLS-1.0</i> at 11-13, 15, 21-23, 41, 46-47, 52-53, 59-60, 71-72, 76-77. In <i>SSL-3.0</i>, MD5 and SHA hash functions of random nonces are performed instead of PRF function. <i>See SSL-3.0</i> at 31 (“md5_hash MD5(master_secret + pad2 + MD5(handshake_messages + Sender + master_secret + pad1)); sha_hash SHA(master_secret + pad2 + SHA(handshake_messages + Sender + master_secret + pad1));”); <i>see also SSL-3.0</i> at 11, 26-27, 30-35, 39, 41, 45-46, 58, 61-62. Fourth, each SSL/TLS protocol discloses or suggests that generating a Certificate Verify message involves the claimed “calculate a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server.” <i>TLS-1.2</i> explains that the client will generate a Certificate Verify message following a client key exchange message, which “compute[s] running hashes for all potential hash algorithms” on all handshake messages sent or received thus far, starting at client hello and up to, but not including, the certificate verify message. <i>TLS-1.2</i> at 62 (Certificate Verify). After receiving the ServerHello message, the client will thus generate a Certificate Verify message that “compute[s] running hashes for all potential hash algorithms” on the exchanged ClientHello and ServerHello messages. <i>Id.</i> Because ClientHello includes ClientHello.random, and because ServerHello includes ServerHello.random, creating this Certificate Verify message after receiving ServerHello also involves calculating a hash function (the hash algorithm) as a function of the random nonce generated by the security engine and a

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	nonce received from the remote server (the ClientHello.random and ServerHello.random, respectively). <i>See also TLS-1.2</i> at 5-6, 16-18, 25-27, 40, 63, 66, 77, 80-81, 84, 92, 94, 97. <i>SSL-3.0, TLS-1.1</i> and <i>TLS-1.0</i> provide similar disclosures as <i>TLS-1.2</i>. <i>See TLS-1.1</i> at 50-51 (“CertificateVerify.signature.md5_hash MD5(handshake_messages); CertificateVerify.signature.sha_hash SHA(handshake_messages)”); <i>see also TLS-1.1</i> at 12-14, 16, 24, 44, 51-53, 59, 66-67, 76, 78, 81-82; <i>TLS-1.0</i> at 45-46 (“CertificateVerify.signature.md5_hash MD5(handshake_messages); CertificateVerify.signature.sha_hash SHA(handshake_messages)”); <i>see also TLS-1.0</i> at 11-13, 15, 21-23, 41, 45-47, 52-53, 59-60, 71-72, 76-77; <i>SSL-3.0</i> at 31 (“CertificateVerify.signature.md5_hash MD5(master_secret + pad_2 + MD5(handshake_messages + master_secret + pad_1)); Certificate.signature.sha_hash SHA(master_secret + pad_2 + SHA(handshake_messages + master_secret + pad_1))”); <i>see also SSL-3.0</i> at 11, 26-27, 30-35, 39, 41, 45-46, 58, 61-62. Additionally, each IKE protocol (<i>RFC-5996</i> or <i>RFC-4306</i>) also teaches to calculate a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server. More specifically, <i>RFC-5996</i> discloses a pseudorandom function PRF “based on HMAC construction.” <i>RFC-5996</i> at 46. Here, HMAC means “Hashed Message Authentication Code.” <i>Id.</i> Therefore, PRF is a hash function. <i>RFC-5996</i> then discloses calculating a series keys, including KEYMAT, SKEYSEED, SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, SK_pr, by calculating a PRF or PRF+ hash function as a function of the random nonce Ni generated by the security engine and a nonce Nr received from the remote server. <i>See RFC-5996</i> at 46-47, 52-53, 11, 13. SKEYSEED and its derivatives are computed as follows: $\text{SKEYSEED} = \text{prf}(\text{Ni} \mid \text{Nr}, g^{\text{ir}})$ $\{\text{SK}_d \mid \text{SK}_{ai} \mid \text{SK}_{ar} \mid \text{SK}_{ei} \mid \text{SK}_{er} \mid \text{SK}_{pi} \mid \text{SK}_{pr}\} = \text{prf+}(\text{SKEYSEED}, \text{Ni} \mid \text{Nr} \mid \text{SPI}_i \mid \text{SPI}_r)$

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>RFC-5996</i> at 47. 2.17. Generating Keying Material for Child SAs A single Child SA is created by the IKE_AUTH exchange, and additional Child SAs can optionally be created in CREATE_CHILD_SA exchanges. Keying material for them is generated as follows: KEYMAT = prf+(SK_d, Ni Nr) Where Ni and Nr are the nonces from the IKE_SA_INIT exchange if this request is the first Child SA created or the fresh Ni and Nr from the CREATE_CHILD_SA exchange if this is a subsequent creation. For CREATE_CHILD_SA exchanges including an optional Diffie-Hellman exchange, the keying material is defined as: KEYMAT = prf+(SK_d, g^ir (new) Ni Nr) where g^ir (new) is the shared secret from the ephemeral Diffie-Hellman exchange of this CREATE_CHILD_SA exchange (represented as an octet string in big endian order padded with zeros in the high-order bits if necessary to make it the length of the modulus). <i>RFC-5996</i> at 52. <i>RFC-4306</i> provides similar disclosures as <i>RFC-5996</i>. See <i>RFC-4306</i> at 8, 10, 28-29, 33-34. Exemplary disclosures from the prior art include the following: <i>Bugiel</i> discloses the following at 225-227 (Establishing SSL/TLS connection and managing two logical SSL/TLS subchannels): Figure 2 illustrates the SSL handshake protocol. The client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters. In the following steps the server's certificate Certs is sent and verified, the SSL/TLS session key Sc is generated (which is never disclosed to WalletHelper) and sealed (together with pre-master secret PMS and the hash value hash(Certs)). The resulting session-token STc

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	is stored in a secure storage and can be loaded and unsealed later using the session identifier SID. ... All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16]. <i>See also Bugiel</i> at 219, 222-230, 232, 234-236. <i>See also Paaske</i> ¶¶ [0054]-[0055], ¶ [0062], ¶ [0105], ¶ [0124], ¶ [0127], ¶ [0131], ¶ [0141], ¶ [0149], ¶ [0153], ¶¶ [0156]-[0157], ¶ [0175], ¶ [0182], ¶ [0191], ¶ [0200], ¶ [0214], Fig. 3. <i>See also Chou</i> ¶¶ [0043]-[0053], ¶¶ [0055]-[0062]. <i>See also Tkacik</i> ¶ [0052], ¶ [0059], ¶ [0067], ¶¶ [0069]-[0070], ¶ [0074], ¶ [0077], ¶¶ [0080]-[0081], ¶ [0083], Fig. 2. <i>See also Tugenberg</i> at 5:47-54. <i>See also Aaraj</i> at 8:5, 8:7, 8:9, 8:11-8:15. <i>See also Gibod</i> ¶¶ [0033]-[0037], ¶ [0071], Fig. 1. <i>See also Mujtaba</i> at ¶ [0054]. <i>See also Buer754</i> at 1:27-41, 1:61-2:8, 5:24-29, 6:13-31, 8:63-9:4, 9:16-21, 13:21-37, 13:45-65, 14:11-34, 14:43-55, 15:4-11, 15:17-16:19, 16:26-35, Fig. 14. <i>See also Weise</i> ¶ [0010], ¶ [0023], ¶¶ [0034]-[0035], claims 9 and 17.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also TLS-1.2</i> at 5-6, 14-18, 25-27, 34, 40, 62-66, 78-82, 84, 92, 94, 97. <i>See also TLS-1.1</i> at 12-14, 16, 24, 32, 44, 50-53, 59, 66-67, 76, 78, 81-82. <i>See also TLS-1.0</i> at 11-13, 15, 21-23, 30, 41, 45-47, 52-53, 59-60, 71-72, 76-77. <i>See also SSL-3.0</i> at 11, 18, 26-27, 30-35, 39, 41, 45-46, 58, 61-62. <i>See also RFC-5996</i> at 46-47, 52-53, 11, 13. <i>See also RFC-4306</i> at 8, 10, 28-29, 33-34. <i>See supra</i> claims 1-4, 5[a].
Claim 6	
[6a] The system-on-a-chip apparatus of claim 1, wherein the security engine is further to generate a public Diffie-Hellman client key and a private Diffie-Hellman client key,	The grounds alleged at the beginning of this chart disclose or suggest “the security engine is further to generate a public Diffie-Hellman client key and a private Diffie-Hellman client key.” The grounds alleged here uses the security engine discussed in [1a-1] to implement <i>Bugiel</i>’s communications method, including to implement known protocols (IPsec IKE or SSL/TLS Handshake Protocol messages), to establish a secure communication. <i>See supra</i> claim 1. In this combination, the security engine discussed in [1a-1] implements the known security protocols (e.g., IPsec IKE or SSL/TLS, such as <i>TLS-1.2</i>) as <i>Bugiel</i> teaches, including the teaching in to perform cryptographic key exchange to generate keys according to the secure communication protocol. <i>See Bugiel</i> at 220 (SSL), 222-234. For example, <i>Chou</i> teaches that HSM 20 (part of the claimed “security engine”) includes a PKA module 56 that provides acceleration for algorithms like “Diffie-Hellman public value and shared key computation” and “Elliptic curve Diffie-Hellman in prime field GF(p).” <i>Chou</i> ¶¶ [0043]-[0053]. In addition, at the time of the '838 patent, generating keys using Diffie-Hellman key exchange was well-known to a POSITA. For example, <i>Paaske</i> similarly teaches that the security engine (e.g., TPM) includes a public key core 350 within the cryptographic processor(s) 330 for performing

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	“Diffie-Hellman.” <i>Paaske</i> ¶ [0055]; <i>Tkacik</i> ¶¶ [0079]-[0080] (“keys may be generated using any of a number of protocols known to those skilled in the art, including IKE, DH, RSA key gen., SSL, random key, and the like.”); <i>Tugenberg</i> at 5:47-54 (SSL); <i>Gibod</i> ¶ [0071] (Diffie-Hellman public key, Diffie-Hellman private key, TLS); <i>Weise</i> ¶ [0010] (SSL and IPsec), ¶ [0023], ¶¶ [0034]-[0035]; <i>Mujtaba</i> ¶ [0006], ¶¶ [0052]-[0053], ¶¶ [0055]-[0057], claim 10, Fig. 7; <i>Buer754</i> at 1:27-41 (IPsec, IKE, SSL), 13:14-15:2 (Diffie-Hellman), 1:61-2:8, 5:24-29, 6:13-31, 8:63-9:4, 9:16-21, 15:17-16:35; <i>Aaraj</i> at 8:9 (SSL), 8:11, 8:15, 8:31. <i>See also</i> [1a-1], [1c]-[1d]. Each of the IKE or SSL/TLS protocols teaches generating a public Diffie-Hellman client key and a private Diffie-Hellman client key. For example, <i>TLS-1.2</i>, which is incorporated by reference in <i>Bugiel</i> (<i>see Bugiel</i> at 225-226, 236) explains that during a TLS key exchange message using Diffie-Hellman key exchange, the client sends the “client’s Diffie-Hellman public value (Yc),” i.e., the claimed public Diffie-Hellman client key, to the server. <i>TLS-1.2</i> at 61 (Client Diffie-Hellman Public Value). <i>TLS-1.2</i> also teaches generating a private Diffie-Hellman client key by “generating a fresh DH private key (X) for each handshake.” <i>TLS-1.2</i> at 93, 87 (cautioning to use a strong and properly seeded random number generator for generating Diffie-Hellman private values); <i>see also TLS-1.2</i> at 9, 47, 49-53, 56-58, 61-62, 65, 72-76, 83, 86-87, 91-93, 100. And <i>TLS-1.2</i> expressly suggests implementing Diffie_Hellman key exchange according to a reference called <i>RFC2785</i>, “<i>Methods for Avoiding the ‘Small-Subgroup’ Attacks on the Diffie-Hellman Key Agreement Method for S/MIME</i>.” <i>See TLS-1.2</i> at 93, 100; <i>see also TLS-1.1</i> at 77, 83. <i>RFC2785</i> explains that, during Diffie-Hellman Key Exchange between Party A and Party B, the parties A and B each generates a private key (xa and xb) and then generates a public key (ya and yb) based on its private key:

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<pre> ya is Party A's public key; ya = g ^ xa mod p yb is Party B's public key; yb = g ^ xb mod p xa is Party A's private key; xa is in the interval [2, (q - 2)] xb is Party B's private key; xb is in the interval [2, (q - 2)] p is a large prime g = h^((p-1)/q) mod p, where h is any integer with 1 < h < p-1 such that h^((p-1)/q) mod p > 1 (g has order q mod p) q is a large prime j a large integer such that p=q*j + 1 </pre> <i>RFC2785</i> at 2. When Party A is client and Party B is server, xa is the claimed “private Diffie-Hellman client key” and ya is the claimed “public Diffie-Hellman client key.” <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provide similar disclosures as <i>TLS-1.2</i>. For example, <i>SSL-3.0</i> describes conveying “client’s Diffie-Hellman public value (Yc)” in client key exchange message. <i>SSL-3.0</i> at 28-30; <i>see also SSL-3.0</i> at 56 (Diffie Hellman private values), 25-27, 30, 32-33, 39-42, 50, 55-57, 60-61; <i>TLS-1.0</i> at 45 (“client’s Diffie-Hellman public value (Yc)”), 70 (Diffie-Hellman private values), 38, 40-41, 43, 48, 52-55, 69-71, 75; <i>TLS-1.1</i> at 50 (“client’s Diffie-Hellman public value (Yc)”), 75 (Diffie-Hellman private values), 41-44, 47, 50, 53, 58-61, 68, 70, 75-77, 83. <i>SSL-3.0</i> also incorporate by reference the original Diffie-Hellman key exchange patents U.S. Patent Nos. 4,200,770 (“<i>Hellman-770</i>”)¹⁸ and 4,218,582 (“<i>Hellman-582</i>”)¹⁹. <i>See SSL-3.0</i> at 60. <i>Hellman-770</i> also describes in a communication system between converser 11 and converser 12, the conversers 11 and 12 each generate a secret key X1 and X2, and derive a public key Y1 and Y2, and then both derive a shared key K by exchanging public keys Y1 and Y2. <i>See Hellman-770</i> at 3:40-5:48, Fig. 1. When

¹⁸ U.S. Patent No. 4,200,770 to Hellman et al. (“*Hellman-770*”).

¹⁹ U.S. Patent No. 4,218,582 to Hellman et al. (“*Hellman-582*”).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	converser 11 is client and converser 12 is server, secret key X1 would be the private Diffie-Hellman client key, public key Y1 would be the public Diffie-Hellman client key, secret key X2 would be the private Diffie-Hellman server key, public key Y2 would be the public Diffie-Hellman server key. <i>Hellman-582</i> also teaches a transmitter and a receiver each generates a pair of keys E and D, where E is public and D is kept secret. <i>See Hellman-582</i> at 1:47-4:47, Fig. 1. When the transmitter is the client and receiver is the server, the public key E of the transmitter would be public Diffie-Hellman client key, the secret key D of the transmitter would be private Diffie-Hellman client key, the public key E of the receiver would be public Diffie-Hellman server key, the secret key D of the receiver would be private Diffie-Hellman server key. <i>See, e.g., Callaway Golf Co. v. Acushnet Co.</i>, 576 F.3d 1331, 1346 (Fed. Cir. 2009) (“To incorporate matter by reference, a host document must contain language ‘clearly identifying the subject matter which is incorporated and where it is to be found.’” (quoting <i>In re De Seversky</i>, 474 F.2d 671, 674 (CCPA 1973))). Each IKE protocol (<i>RFC-5996</i> or <i>RFC-4306</i>) also teaches to generate a public Diffie-Hellman client key and a private Diffie-Hellman client key. More specifically, <i>RFC-5996</i> discloses in the initial exchanges, “[t]he first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange [DH].” <i>RFC-5996</i> at 9; <i>see also RFC-5996</i> at 6, 9-16, 22, 33, 44-48, 52-53, 76, 80-83, 86-87, 121, 126, 128, 130-131, 133, 134, 137. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 7 (“The first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange [DH].”); <i>see also RFC-4306</i> at 4, 7-10, 21, 26-30, 33-34, 47, 50-52, 55-56, 88-89, 91-92, 95, 97. In addition, <i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>RFC-5996</i> and <i>RFC-4306</i> all expressly incorporate by reference the original Diffie-Hellman reference “New Directions in Cryptography” (“<i>Diffie</i>”).²⁰ <i>See SSL-3.0</i> at 25, 60-61; <i>TLS-1.0</i> at 75; <i>RFC-5996</i> at 9, 128; <i>RFC-4306</i> at 7, 92. <i>Diffie</i> teaches that user i and user j each

²⁰ W. Diffie & M.E. Hellman, “New Directions in Cryptography,” IEEE Transactions on Information Theory, vol. IT-22, no. 6, pp. 74-84 (June 1977) (“*Diffie*”).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	generates public Diffie-Hellman key Y_i, Y_j and a private Diffie-Hellman key X_i, X_j to derive a shared key K_{ij}. <i>See Diffie</i> at p. 649. When user i is client and user j is server, Y_i would be the claimed public Diffie-Hellman client key, X_i would be the claimed private Diffie-Hellman client key, Y_j would be the public Diffie-Hellman server key, X_j would be the private Diffie-Hellman server key. <i>See Diffie</i> at pp. 644-645 (describing a transmitter, a receiver and an eavesdropper in a communication system), Figs. 1-2. as their key. User i obtains K_{ij} by obtaining Y_j from the public file and letting $K_{ij} = Y_j^{X_i} \bmod q \quad (9)$ $= (\alpha^{X_j})^{X_i} \bmod q \quad (10)$ $= \alpha^{X_j X_i} = \alpha^{X_i X_j} \bmod q. \quad (11)$ User j obtains K_{ij} in the similar fashion $K_{ij} = Y_i^{X_j} \bmod q. \quad (12)$ Each user generates an independent random number X_i chosen uniformly from the set of integers $\{1, 2, \dots, q - 1\}$. Each keeps X_i secret, but places Another user must compute K_{ij} from Y_i and Y_j, for example, by computing $K_{ij} = Y_i^{(\log_\alpha Y_j)} \bmod q. \quad (13)$ $Y_i = \alpha^{X_i} \bmod q \quad (7)$ in a public file with his name and address. When users i and j wish to communicate privately, they use $K_{ij} = \alpha^{X_i X_j} \bmod q \quad (8)$ <i>Diffie</i> at pp. 649. The prior art grounds alleged here thus disclose or suggest “the security engine is to generate a private Diffie-Hellman client key and then generating a public Diffie-Hellman client key based on the private Diffie-Hellman client key,” as claimed.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Exemplary disclosures of the prior art include the following: <i>See Bugiel</i> at 220, 222-234. <i>See also Paaske</i> ¶ [0055]. <i>See also Chou</i> ¶¶ [0043]-[0053]. <i>See also Tkacik</i> ¶¶ [0079]-[0080]. <i>See also Tugenberg</i> at 5:47-54. <i>See also Gibod</i> ¶ [0071]. <i>See also Weise</i> ¶ [0010], ¶ [0023], ¶¶ [0034]-[0035]. <i>See also Mujtaba</i> ¶ [0006], ¶¶ [0052]-[0053], ¶¶ [0055]-[0057], claim 10, Fig. 7. <i>See also Buer754</i> at 1:27-41, 13:14-15:2, 1:61-2:8, 5:24-29, 6:13-31, 8:63-9:4, 9:16-21, 15:17-16:35. <i>See also Aaraj</i> at 8:9, 8:11, 8:15, 8:31. <i>See also TLS-1.2</i> at 9, 47, 49-53, 56-58, 61-62, 65, 72-76, 83, 86-87, 91-93, 100. <i>See also SSL-3.0</i> at 25-30, 32-33, 39-42, 50, 55-57, 60-61. <i>See also TLS-1.0</i> at 38, 40-41, 43, 45, 48, 52-55, 69-71, 75. <i>See also TLS-1.1</i> at 41-44, 47, 50, 53, 58-61, 68, 70, 75-77, 83.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also RFC-5996</i> at 6, 9-16, 22, 33, 44-48, 52-53, 76, 80-83, 86-87, 121, 126, 128, 130-131, 133-134, 137. <i>See also RFC-4306</i> at 4, 7-10, 21, 26-30, 33-34, 47, 50-52, 55-56, 88-89, 91-92, 95, 97. <i>See also</i> claim 1.
[6b] sign the public Diffie-Hellman client key using the security key of the security engine, and send the signed public Diffie-Hellman key to the remote server.	The grounds alleged at the beginning of this chart disclose or suggest to “sign the public Diffie-Hellman client key using the security key of the security engine, and send the signed public Diffie-Hellman key to the remote server.” The grounds alleged here uses the security engine discussed in [1a-1] and security key discussed in [1a-3] to implement <i>Bugiel</i>’s communications method in the secure execution environment based on known protocols (e.g., IPsec IKE or SSL/TLS Handshake Protocol messages, etc.), to establish a secure communication. In the combinations, the security engine discussed in [1a-1] implements one of IKE or SSL/TLS protocols as <i>Bugiel</i> teaches, including the teaching in the IKE and SSL/TLS protocols to perform cryptographic key exchange. <i>See Bugiel</i> at 220 (SSL/TLS), 219 (authentication), 222-234, 236. In addition, at the time of the '838 patent, digital signature for authentication was common knowledge to a POSITA. For example, <i>Chou</i> explains that PKA module 56 (part of the claimed “security engine”) performs digital signing using 1024-bit DSA (Digital Signature Algorithm) signature. <i>Chou</i> ¶¶ [0043]-[0053], ¶ [0003] (keys for authentication), ¶ [0005], ¶ [0012], ¶ [0017], ¶ [0021], ¶ [0038], ¶ [0040], ¶ [0069]; <i>Paaske</i> ¶ [0005] (“Cryptographic techniques may include one or more of . . . signing . . .”), ¶¶ [0028]-[0031], ¶¶ [0041]-[0042], ¶ [0055], ¶ [0066], ¶ [0106], ¶¶ [0167]-[0170], ¶ [0175], ¶ [0177], ¶¶ [0179]-[0180], ¶¶ [0193]-[0194], ¶ [0194], ¶¶ [0204]-[0212], ¶ [0215], claim 25, Fig. 3; <i>Tugenberg</i> at 5:47-54 (SSL), 4:32-36 (“the user’s private key may be used for authentication of digital signatures as well as decryption of data”); <i>Tkacik</i> ¶¶ [0079]-[0080], ¶ [0046], ¶ [0009] (authentication), ¶ [0051]; <i>Mujtaba</i> ¶ [0021] (sign), ¶ [0031], ¶¶ [0034]-[0035], claims 6, 15, 22; <i>Gibod</i> ¶ [0071] (TLS), ¶¶ [0007]-[0008] (signature verification), ¶ [0019], ¶¶ [0024]-[0025], ¶¶ [0027]-[0031], ¶¶ [0033]-[0037], ¶ [0039], ¶ [0056], ¶ [0067], claims 4, 8-9; <i>Buer754</i> at 1:26-41 (SSL, IPsec, IKE), 13:14-16:35 (RSA authenticate, DSA Sign, SSL, IPsec, a variety of

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	cryptographic and signature algorithms and modifications), 1:61-2:8, 5:24-29, 6:13-35, 7:42-58, 8:35-50, 8:63-9:4, 9:16-21, 10:29-41, 11:4-16; <i>Aaraj</i> at 8:9 (SSL), 8:5-8:6 (RSA signing), 8:8-8:11, 8:13-8:16, 8:20, 8:27-8:28, 8:31; <i>Weise</i> at ¶ [0010] (SSL, IPsec), ¶ [0023], ¶¶ [0033]-[0035], Fig. 3, claims 9 and 17; <i>see supra</i> [1a-1] (disclosures from each reference on security engine). <i>Chou</i> and other references also disclose a “security key of the security engine” as discussed in [1a-3]. <i>See supra</i> [1a-3], [1e] (disclosures from each reference on security key). The SSL/TLS Handshake protocol teaches signing the public Diffie-Hellman client key and send the signed public Diffie-Hellman key to the remote server. For example, <i>TLS-1.2</i> teaches that the public Diffie-Hellman client key Yc (the claimed “public Diffie-Hellman client key”) needs to be sent to the server so the server can authenticate the client and verify the client has corresponding private key. <i>See TLS-1.2</i> at 57-62 (showing the client key exchange message includes the public Diffie-Hellman client key ClientDiffieHellmanPublic or dh_Yc); <i>see also TLS-1.2</i> at 74. <i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i> provide similar disclosures. <i>See SSL-3.0</i> at 28-30 (showing the client key exchange message includes public Diffie-Hellman client key ClientDiffieHellmanPublic or dh_Yc), 40-41; <i>TLS-1.0</i> at 43-45 (showing the client key exchange message includes public Diffie-Hellman client key ClientDiffieHellmanPublic or dh_Yc), 53-54; <i>TLS-1.1</i> at 47-50 (showing the client key exchange message includes public Diffie-Hellman client key ClientDiffieHellmanPublic or dh_Yc), 59-60. <i>TLS-1.2</i> also teaches signing the Diffie-Hellman public client key before sending it to the server. <i>See TLS-1.2</i> at 62 (Certificate Verify), 75, 78. More specifically, <i>TLS-1.2</i> teaches sending a Certificate Verify message after the client key exchange message. The Certificate Verify message includes “digitally-signed” version of all previous handshake messages, which would include the client key exchange message that contains public Diffie-Hellman client key. According to <i>TLS-1.2</i>, the purpose of sending the Certificate Verify message is to “provide explicit verification of a client certificate.” <i>Id.</i> at 62 (Certificate Verify), 33-37 (Handshake Protocol includes sending the certificate verify message), 74.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<pre> Structure of this message: struct { digitally-signed struct { opaque handshake_messages[handshake_messages_length]; } } CertificateVerify; </pre> <i>TLS-1.2</i> at 62. <i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i> provide similar disclosures. <i>See SSL-3.0</i> at 18-20 (Handshake Protocol includes sending the certificate verify message), 30-31 (Certificate verify), 40-41; <i>TLS-1.0</i> at 29-32 (Handshake Protocol includes sending the certificate verify message), 45-46 (Certificate verify), 53-55; <i>TLS-1.1</i> at 31-34 (Handshake Protocol includes sending the certificate verify message), 50-51 (Certificate verify), 59-61. The IKE protocols (<i>RFC-5996</i> or <i>RFC-4306</i>) also teach sending the Diffie-Hellman public client key to the remote server. More specifically, <i>RFC-5996</i> discloses in the initial exchanges, “[t]he first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange [DH].” <i>RFC-5996</i> at 9; <i>see also RFC-5996</i> at 6, 9-16, 22, 33, 44-48, 52-53, 76, 80-83, 86-87, 121, 126, 128, 130-131, 133, 134, 137. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 7 (“The first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange [DH].”); <i>see also RFC-4306</i> at 4, 7-10, 21, 26-30, 33-34, 47, 50-52, 55-56, 88-89, 91-92, 95, 97. In addition, <i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i>, <i>TLS-1.2</i>, <i>RFC-5996</i> and <i>RFC-4306</i> all expressly incorporate by one or more Diffie-Hellman related references, such as <i>RFC2785</i>, <i>Hellman-770</i>, <i>Hellman-582</i>, or <i>Diffie</i> respectively, which all describe sending the public Diffie-Hellman client key to the remote server. <i>See</i> [6a]. In addition, the IKE and SSL/TLS protocols suggest using a key specific to the client device so the server could verify and authenticate the signed public Diffie-Hellman client key based on a device-specific key. <i>See TLS-1.2</i> at 33-37 (“Exchange certificates and cryptographic information to allow the

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	client and server to authenticate themselves . . . and to explicitly verify possession of the private key in the certificate.”), 47-50, 55-57; <i>see also</i> <i>SSL-3.0</i> at 30 (“to provide explicit verification of a client certificate”), 25, 28; <i>TLS-1.0</i> at 29-32 (“Exchange certificates and cryptographic information to allow the client and server to authenticate themselves a digital-signed certificate verify message is sent to explicitly verify the certificate.”), 37-39, 43; <i>TLS-1.1</i> at 31-34 (“Exchange certificates and cryptographic information to allow the client and server to authenticate themselves a digital-signed certificate verify message is sent to explicitly verify the certificate.”), 40-42, 46; <i>RFC-5996</i> at 12 (“Both parties in the IKE_AUTH exchange MUST verify that all signatures and Message Authentication Codes (MACs) are computed correctly”), 47-50, 95-96, 120; <i>RFC-4306</i> at 9 (“The recipients of messages 3 and 4 MUST verify that all signatures and MACs are computed correctly”), 29-31, 63. One of the most obvious keys for this purpose would have been the security key that is encoded during manufacturing of the client device as discussed in [1a-3]. <i>See supra</i> [1a-3], [1f]. For example, Because <i>Chou</i> teaches a security key encoded in the secure memory when manufacturing the client device, the security key is a device-specific key that a POSITA would have found obvious to use for signing the public Diffie-Hellman client key so the server can verify the identity of the client device. <i>See Chou</i> ¶ [0038] (explaining “OTP 54 is a non-volatile memory that is used to one-time program a key value or values for device 30. For example, the device unique key may be programmed and stored in OTP 54.”). In addition to <i>Chou</i>, other references also teach the security key as discussed in [1a-3]. <i>See also supra</i> [1a-3], [1f] (disclosure of security key in the prior art references). A POSITA thus would have found claim feature [6b] obvious based on the grounds alleged here. Exemplary disclosures of the prior art include the following: <i>See also TLS-1.2</i> at 33-37, 47-50, 55-62, 74-75, 78. <i>See also SSL-3.0</i> at 18-20, 25, 28-31, 40-41. <i>See also TLS-1.0</i> at 29-32, 37-39, 43-45, 53-55.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also TLS-I.1</i> at 31-34, 40-42, 46-51, 59-61. <i>See also RFC-5996</i> at 6, 9-16, 22, 33, 44-50, 52-53, 76, 80-83, 86-87, 95-96, 120-121, 126, 128, 130-131, 133, 134, 137. <i>See also RFC-4306</i> at 4, 7-10, 21, 26-31, 33-34, 47, 50-52, 55-56, 63, 88-89, 91-92, 95, 97. <i>See also Paaske</i> ¶ [0005], ¶¶ [0028]-[0031], ¶¶ [0041]-[0042], ¶¶ [0055]-[0056], ¶ [0066], ¶ [0101], ¶ [0106], ¶ [0108], ¶¶ [0167]-[0170], ¶ [0175], ¶ [0177], ¶¶ [0179]-[0180], ¶¶ [0193]-[0194], ¶¶ [0204]-[0212], ¶ [0215], claim 25, Fig. 3. <i>See also Tugenberg</i> at 4:32-36, 5:47-54. <i>See also Chou</i> ¶¶ [0043]-[0053], ¶ [0003], ¶ [0005], ¶ [0012], ¶ [0017], ¶ [0021], ¶ [0038], ¶ [0040], ¶ [0069]. <i>See also Bugiel</i> at 219-220, 222-234, 236. <i>See also Tkacik</i> ¶¶ [0079]-[0080], ¶ [0046], ¶ [0009], ¶ [0051]. <i>See also Mujtaba</i> ¶ [0021], ¶ [0031], ¶¶ [0034]-[0035], claims 6, 15, 22. <i>See also Gibod</i> ¶ [0071], ¶¶ [0007]-[0008], ¶ [0019], ¶¶ [0024]-[0025], ¶¶ [0027]-[0031], ¶¶ [0033]-[0037], ¶ [0039], ¶ [0056], ¶ [0067], claims 4, 8-9. <i>See also Buer754</i> at 1:26-41, 13:14-16:35, 1:61-2:8, 5:24-29, 6:13-35, 7:42-58, 8:35-50, 8:63-9:4, 9:16-21, 10:29-41, 11:4-16. <i>See also Aaraj</i> at 8:5-8:6, 8:8-8:11, 8:13-8:16, 8:20, 8:27-8:28, 8:31. <i>See also Weise</i> at ¶ [0010], ¶ [0023], ¶¶ [0033]-[0035], Fig. 3, claims 9 and 17.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See supra</i> Claims 1, [6a].
Claim 7	
[7] The system-on-a-chip apparatus of claim 6, wherein the security engine is further to receive a public Diffie-Hellman server key from the remote server and generate the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key.	The grounds alleged at the beginning of this chart disclose or suggest “the security engine is further to receive a public Diffie-Hellman server key from the remote server and generate the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key.” The grounds alleged here uses the security engine discussed in [1a-1] to implement <i>Bugiel</i>’s communications method in <i>Bugiel</i>’s secure execution environment, including to implement known protocols, e.g., the IPsec IKE or SSL/TLS Handshake Protocol messages, etc., to establish a secure communication. In this combination, the security engine discussed in [1a-1] implements IPsec IKE or SSL/TLS protocols as <i>Bugiel</i> teaches, including the teaching in IPsec or SSL/TLS to receive a public Diffie-Hellman server key from the remote server and to generate a session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key. For example, <i>Chou</i> teaches that HSM 20 (part of the claimed “security engine”) includes a PKA module 56 that provides acceleration for algorithms like “Diffie-Hellman public value and shared key computation” and “Elliptic curve Diffie-Hellman in prime field GF(p).” <i>Chou</i> ¶¶ [0043]-[0053]; <i>see also supra</i> [1a-1], [1c]-[1d] (relevant disclosures from each reference) Each IKE and SSL/TLS protocol teaches “receiving a public Diffie-Hellman server key from the remote server.” <i>TLS-1.2</i> explains that the client device receives a public Diffie-Hellman server key dh_Ys (the claimed “public Diffie-Hellman server key”) from the remote server to complete a key exchange and to calculate the premaster secret. <i>TLS-1.2</i> at 50-53 (Server key exchange message); <i>see also TLS-1.2</i> at 47-50, 72-73, 65, 75, 86, 92-93, 100. <i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i> also provide similar disclosures. <i>See SSL-3.0</i> at 25-27 (receiving dh_Ys in server key exchange message), 28, 33, 39-41, 57, 60-61; <i>TLS-1.0</i> at 39-41 (receiving dh_Ys in server key exchange message), 37-39, 43, 48, 52-55, 71, 75; <i>TLS-1.1</i> at 42-44 (receiving dh_Ys in server key exchange message), 40-42, 46-47, 53, 61, 76-77.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Each IKE protocol (<i>RFC-5996</i> or <i>RFC-4306</i>) also teaches receiving a public Diffie-Hellman server key from the remote server. More specifically, <i>RFC-5996</i> discloses in the initial exchanges, “[t]he first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange [DH].” <i>RFC-5996</i> at 9; <i>see also RFC-5996</i> at 6, 9-16, 22, 33, 44-48, 52-53, 76, 80-83, 86-87, 121, 126, 128, 130-131, 133, 134, 137. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 7 (“The first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange [DH].”); <i>see also RFC-4306</i> at 4, 7-10, 21, 26-30, 33-34, 47, 50-52, 55-56, 88-89, 91-92, 95, 97. In addition, <i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i>, <i>TLS-1.2</i>, <i>RFC-5996</i> and <i>RFC-4306</i> all expressly incorporate by one or more Diffie-Hellman related references, such as <i>RFC2785</i>, <i>Hellman-770</i>, <i>Hellman-582</i>, or <i>Diffie</i> respectively, which all receiving a public Diffie-Hellman server key from the remote server to calculate a shared key. <i>See</i> claim 6. Each IKE and SSL/TLS protocol teaches to “generate the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key” in two ways. Before discussing the two ways each SSL/TLS protocol satisfies this portion of claim 7, first note the premaster secret is a function of the public Diffie-Hellman server key and private Diffie-Hellman client key. <i>TLS-1.2</i> teaches that the client device generates the premaster secret by performing conventional Diffie-Hellman using the public Diffie-Hellman server key received from the server and private Diffie-Hellman client key. <i>See TLS-1.2</i> at 50-53 (explaining the result of Diffie-Hellman key exchange is the “premaster secret”), 65 (Diffie-Hellman); <i>see also TLS-1.2</i> at 34, 55-58, 91-93, 100. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.2</i> provide similar disclosures. <i>See SSL-3.0</i> at 57 (“the client and server will generate the same Diffie-Hellman result (i.e., pre_master_secret) every time they communicate.”), 33 (Diffie-Hellman), 25-27, 56, 61; <i>TLS-1.0</i> at 39-41 (explaining the result of Diffie-Hellman key exchange is the “premaster secret”), 71 (“the client and server will generate the same Diffie-Hellman result (i.e., pre_master_secret) every time they communicate.”), 48 (Diffie-Hellman), 75, 29, 43-44; <i>TLS-1.1</i> at 42-44 (explaining the result of Diffie-Hellman key exchange is the “premaster secret”), 53

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	(Diffie-Hellman), 76 (“the client and server will generate the same Diffie-Hellman result (i.e., pre_master_secret) every time they communicate.”), 31, 47, 75-77, 83. In addition, both <i>TLS-1.2</i> and <i>TLS-1.1</i> suggest using the teachings in <i>RFC2785</i>, “<i>Methods for Avoiding the ‘Small-Subgroup’ Attacks on the Diffie-Hellman Key Agreement Method for S/MIME</i>,” to implement Diffie-Hellman when calculating the premaster secret. See <i>TLS-1.2</i> at 93, 100; <i>TLS-1.1</i> at 77, 83. <i>RFC2785</i>, in turn, teaches that a shared key (ZZ), i.e., the pre-master key, is generated based on the calculation below. <i>RFC2785</i> at 2. <pre style="margin-left: 40px;"> Note that the individual parties actually perform the computations: ZZ = (yb ^ xa) mod p = (ya ^ xb) mod p where ^ denotes exponentiation. ya is Party A's public key; ya = g ^ xa mod p yb is Party B's public key; yb = g ^ xb mod p xa is Party A's private key; xa is in the interval [2, (q - 2)] xb is Party B's private key; xb is in the interval [2, (q - 2)] p is a large prime g = h^((p-1)/q) mod p, where h is any integer with 1 < h < p-1 such that h^((p-1)/q) mod p > 1 (g has order q mod p) q is a large prime j a large integer such that p=q*j + 1 </pre> When Party A is the client device and Party B is the server, <i>RFC2785</i> teaches that the shared key (ZZ), which <i>TLS-1.2</i> and <i>TLS-1.1</i> use as premaster key, is generated as a function of public Diffie-Hellman server key yb and private Diffie-Hellman client key xa (specifically, $ZZ = (yb ^ xa) \text{ mod } p$). <i>TLS-1.2</i> and <i>TLS-1.1</i> thus teaches generating the premaster secret as a function of public Diffie-Hellman server key yb and private Diffie-Hellman client key xa. In addition, <i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>RFC-5996</i> and <i>RFC-4306</i> all expressly incorporate by reference the original Diffie-Hellman reference <i>Diffie</i>. See <i>SSL-3.0</i> at 25, 60-61; <i>TLS-1.0</i> at 75; <i>RFC-5996</i> at 9, 128; <i>RFC-4306</i> at 7, 92. <i>Diffie</i> teaches that user i and user j each exchange public Diffie-Hellman key Yi and Yj with each other, and derives a shared key Kij from its own private Diffie-Hellman key and the other's public Diffie-Hellman key. See <i>Diffie</i> at p. 649. When user i is client and user j is server, Yi

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	would be the claimed public Diffie-Hellman client key, X_i would be the claimed private Diffie-Hellman client key, Y_j would be the public Diffie-Hellman server key, X_j would be the private Diffie-Hellman server key, and K_{ij} would be the claimed premaster key. <i>See Diffie</i> at pp. 644-645 (describing a transmitter, a receiver and an eavesdropper in a communication system), Figs. 1-2. Therefore, each of <i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>RFC-5996</i> and <i>RFC-4306</i> also teach generating the premaster secret K_{ij} as a function of public Diffie-Hellman server key Y_j and private Diffie-Hellman client key X_i through their incorporation of <i>Diffie</i>. as their key. User i obtains K_{ij} by obtaining Y_j from the public file and letting $K_{ij} = Y_j^{X_i} \bmod q \quad (9)$ $= (\alpha^{X_j})^{X_i} \bmod q \quad (10)$ $= \alpha^{X_j X_i} = \alpha^{X_i X_j} \bmod q. \quad (11)$ User j obtains K_{ij} in the similar fashion $K_{ij} = Y_i^{X_j} \bmod q. \quad (12)$ Each user generates an independent random number X_i chosen uniformly from the set of integers $\{1, 2, \dots, q-1\}$. Each keeps X_i secret, but places $Y_i = \alpha^{X_i} \bmod q \quad (7)$ in a public file with his name and address. When users i and j wish to communicate privately, they use $K_{ij} = \alpha^{X_i X_j} \bmod q \quad (8)$ Another user must compute K_{ij} from Y_i and Y_j, for example, by computing $K_{ij} = Y_i^{(\log_\alpha Y_j)} \bmod q. \quad (13)$ We thus see that if logs mod q are easily computed the system can be broken. While we do not currently have a proof of the converse (i.e., that the system is secure if logs mod q are difficult to compute), neither do we see any way to compute K_{ij} from Y_i and Y_j without first obtaining either X_i or X_j. <i>Diffie</i> at pp. 649. <i>SSL-3.0</i> also incorporate by reference the original Diffie-Hellman key exchange patents U.S. Patent <i>Hellman-770</i>. <i>See SSL-3.0</i> at 60. <i>Hellman-770</i> describes in a communication system between converser 11 and converser 12, the conversers 11 and 12 each generate a secret key X_1 and X_2, and a public key Y_1 and Y_2, and then both derive a shared key K by exchanging public keys Y_1 and Y_2. <i>See Hellman-770</i> at 3:40-5:48, Fig. 1. When converser 11 is client and converser 12 is server, secret key

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	X1 would be the private Diffie-Hellman client key, public key Y1 would be the public Diffie-Hellman client key, secret key X2 would be the private Diffie-Hellman server key, public key Y2 would be the public Diffie-Hellman server key, and the shared key K would be the claimed premaster key. Therefore, <i>SSL-3.0</i> also teaches generating the premaster secret K as a function of public Diffie-Hellman server key Y2 and private Diffie-Hellman client key X1 through its incorporation of <i>Hellman-770</i>. In sum, each of the SSL/TLS protocols teaches generating the premaster key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key. Turning to the two ways each of the SSL/TLS protocols satisfies the second part of claim 7, <u>first, each of the SSL/TLS protocols teaches generating the master secret (the claimed “session key”) as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key.</u> When the master secret is mapped to the claimed “session key,” each of the SSL/TLS protocols satisfies the second part of claim 7 by teaching (1) generating the master secret (the claimed “session key”) as a function of pre-master secret and (2) generating a premaster secret as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key. For example, <i>TLS-1.2</i> teaches generating the master secret as a function of the pre-master secret. <i>See TLS-1.2</i> at 64 (“master_secret = PRF (pre_master_secret, “master secret”, ClientHello.random + ServerHello.random)”); <i>see also TLS-1.2</i> at 34, 58, 65, 91-93. <i>SSL-3.0</i>, <i>TLS-1.0</i> and <i>TLS-1.1</i> provide similar disclosures. <i>See SSL-3.0</i> at 32 (“master_secret = MD5(pre_master_secret + SHA('A' + pre_master_secret + ClientHello.random + ServerHello.random)) + MD5(pre_master_secret + SHA('BB' + pre_master_secret + ClientHello.random + ServerHello.random)) + MD5(pre_master_secret + SHA('CCC' + pre_master_secret + ClientHello.random + ServerHello.random))”), 29-30, 33, 55, 57; <i>TLS-1.0</i> at 47 (“master_secret = PRF (pre_master_secret, “master secret”, ClientHello.random + ServerHello.random)”), 29, 45, 48, 69, 71; <i>TLS-1.1</i> at 52-53 (“master_secret = PRF (pre_master_secret, “master secret”, ClientHello.random + ServerHello.random)”), 31, 48, 75-76.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	As explained above, in each of the SSL/TLS protocols, the premaster secret is a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key. Because each of the SSL/TLS protocols teaches generating the master secret as a function of the pre-master secret, and the premaster secret is a function of the private Diffie-Hellman client key and public Diffie-Hellman server keys, the master secret (the claimed “session key”) is generated as a function of the public Diffie-Hellman server key and private Diffie-Hellman client key. <u>Second, each of the SSL/TLS protocols teaches generating the key_block (the claimed “session key”) as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key.</u> Each of the SSL/TLS protocols teaches (1) generating key_block (the claimed “session key”) as a function of master secret; (2) generating the master secret as a function of pre-master secret; and (3) generating a premaster secret as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key (as discussed above). For example, <i>TLS-1.2</i> teaches that the key_block (the claimed “session key”) is generated as a function of the master secret. <i>See TLS-1.2</i> at 26 (“key_block = PRF(SecurityParameters.master_secret, ‘key expansion’, SecurityParameters.server_random + SecurityParameters.client_random)”); <i>see also TLS-1.2</i> at 16, 25-26, 58, 80, 84, 91, 94. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> also provide similar disclosures. <i>See SSL-3.0</i> at 34 (“key_block = MD5(master_secret + SHA(`A' + master_secret + ServerHello.random + ClientHello.random)) + MD5(master_secret + SHA(`BB' + master_secret + ServerHello.random + ClientHello.random)) + MD5(master_secret + SHA(`CCC' + master_secret + ServerHello.random + ClientHello.random)) + [...]”), 33-35, 45, 48, 55; <i>TLS-1.0</i> at 21 (“key_block = PRF (SecurityParameters.master_secret, “key expansion”, SecurityParameters.server_random + SecurityParameters.client_random)”), 21-23, 59, 62, 69, 72; <i>TLS-1.1</i> at 24 (“key_block = PRF (SecurityParameters.master_secret, “key expansion”, SecurityParameters.server_random + SecurityParameters.client_random)”), 66, 69, 78. Because the master secret is generated as a function of the pre-master secret (as discussed above), and because the pre-master secret generated as a function of the private Diffie-Hellman client key and public Diffie-Hellman server key (as also discussed above), the key_block (the claimed “session key”)

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	is generated as a function of the private Diffie-Hellman client key and public Diffie-Hellman server key. Each of the SSL/TLS protocols thus teaches generating the key_block (the claimed “session key”) as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key. In addition, each IKE protocol (<i>RFC-5996</i> and <i>RFC-4306</i>) also teaches generating the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key. As explained above, <i>RFC-5996</i> and <i>RFC-4306</i> also teaches generating the premaster secret K_{ij} as a function of public Diffie-Hellman server key Y_j and private Diffie-Hellman client key X_i through their incorporation of <i>Diffie</i>. <i>RFC-5996</i> further denotes the shared premaster g^{ir} or g^{ir} (new) and discloses generating a series of keys such as SKEYSEED, SK_d, SK_{ai}, SK_{ar}, SK_{ei}, SK_{er}, KEYMAT as a function of g^{ir} or g^{ir} (new). <i>See RFC-5996</i> at 47, 52. <pre> SKEYSEED = prf(Ni Nr, g^ir) {SK_d SK_ai SK_ar SK_ei SK_er SK_pi SK_pr } = prf+ (SKEYSEED, Ni Nr SPIi SPIr) </pre> (indicating that the quantities SK_d, SK_{ai}, SK_{ar}, SK_{ei}, SK_{er}, SK_{pi}, and SK_{pr} are taken in order from the generated bits of the $prf+$). g^{ir} is the shared secret from the ephemeral Diffie-Hellman exchange. g^{ir} is represented as a string of octets in big endian order padded with zeros if necessary to make it the length of the modulus. N_i and N_r are the nonces, stripped of any headers. For historical backward-compatibility reasons, there are two PRFs that are treated specially in this calculation. If the negotiated PRF is AES-XCBC-PRF-128 [AESXCBCPRF128] or AES-CMAC-PRF-128 [AESCMACPRF128], only the first 64 bits of N_i and the first 64 bits of N_r are used in calculating SKEYSEED, but all the bits are used for input to the $prf+$ function.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	2.17. Generating Keying Material for Child SAs A single Child SA is created by the IKE_AUTH exchange, and additional Child SAs can optionally be created in CREATE_CHILD_SA exchanges. Keying material for them is generated as follows: KEYMAT = prf+(SK_d, Ni Nr) Where Ni and Nr are the nonces from the IKE_SA_INIT exchange if this request is the first Child SA created or the fresh Ni and Nr from the CREATE_CHILD_SA exchange if this is a subsequent creation. For CREATE_CHILD_SA exchanges including an optional Diffie-Hellman exchange, the keying material is defined as: KEYMAT = prf+(SK_d, g^ir (new) Ni Nr) where g^ir (new) is the shared secret from the ephemeral Diffie-Hellman exchange of this CREATE_CHILD_SA exchange (represented as an octet string in big endian order padded with zeros in the high-order bits if necessary to make it the length of the modulus). <i>RFC-5996</i> at 47, 52; <i>see also RFC-5996</i> at 9-13, 46-47, 52-53. Under Daedalus' apparent view, any of these keys could be the symmetric session key. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 7-10, 28-29, 33-34; <i>see also</i> claim features [1c]-[1d] and [5]. Exemplary disclosures of the prior art include the following: <i>See also TLS-1.2</i> at 16, 25-26, 34, 47-53, 55-58, 64-65, 72-73, 75, 80, 84, 86, 91-94, 100. <i>See also SSL-3.0</i> at 25-30, 32-35, 39-41, 45, 48, 55-57, 60-61. <i>See also TLS-1.0</i> at 21-23, 29, 37-41, 43-45, 47-48, 52-55, 59, 62, 69, 71-72, 75. <i>See also TLS-1.1</i> at 24, 31, 40-44, 46-48, 52-53, 61, 66, 69, 75-78, 83.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also RFC-5996</i> at 6, 9-16, 22, 33, 44-48, 52-53, 76, 80-83, 86-87, 121, 126, 128, 130-131, 133-134, 137. <i>See also RFC-4306</i> at 4, 7-10, 21, 26-30, 33-34, 47, 50-52, 55-56, 88-89, 91-92, 95, 97. <i>See also supra</i> claims 1, 5, 6.
Claim 8	
[8] The system-on-a-chip apparatus of claim 1, wherein the security engine is further to receive Diffie-Hellman global values and a public Diffie-Hellman server key from the remote server and store at least one of the Diffie-Hellman global values and the public Diffie-Hellman server key in the secure memory.	The grounds alleged at the beginning of this chart disclose or suggest “the security engine is further to receive Diffie-Hellman global values and a public Diffie-Hellman server key from the remote server and store at least one of the Diffie-Hellman global values and the public Diffie-Hellman server key in the secure memory.” The grounds alleged here use the security engine discussed in [1a-1] to implement <i>Bugiel</i>’s communications method in <i>Bugiel</i>’s secure execution environment, including to implement IPsec IKE or SSL/TLS Handshake Protocol messages, to establish a secure communication. In this combination, the security engine discussed in [1a-1] implements IPsec IKE or SSL/TLS Handshake, e.g., TLS version 1.2 as <i>Bugiel</i> teaches, including the teaching to receive Diffie-Hellman global values and a public Diffie-Hellman server key from the remote server and to store at least one of the Diffie-Hellman global values and the public Diffie-Hellman server key. In addition, at the time of the '838 patent, it was well-known to a POSITA to receive Diffie-Hellman global values and a public Diffie-Hellman server key from the remote server and store at least one of the Diffie-Hellman global values and the public Diffie-Hellman server key in the secure memory using the security engine as part of the Diffie-Hellman key exchange process. As one example, <i>Chou</i> teaches that HSM 20 (part of the claimed “security engine”) includes a PKA module 56 that provides acceleration for algorithms like “Diffie-Hellman public value and shared key computation” and “Elliptic curve Diffie-Hellman in prime field GF(p).” <i>Chou</i> ¶¶ [0043]-[0053]. In addition to <i>Chou</i>, other references also teach the security engine is capable of performing relevant cryptographic functions. <i>See supra</i> [1a-1], [1c]-[1d], [6]-[7].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	The grounds alleged here teach “the security engine is further to receive Diffie-Hellman global values and a public Diffie-Hellman server key from the remote server.” To calculate the premaster secret, <i>TLS-1.2</i> teaches receiving Diffie-Hellman global values <i>dh_p</i> (prime) and <i>dh_g</i> (primitive root modulo <i>p</i>) (together forming the claimed “Diffie-Hellman global values”), as well as <i>DH_Ys</i> (the claimed “public Diffie-Hellman server key”), in the server key exchange message from the server. <i>TLS-1.2</i> at 50-53 (Server Key Exchange Message); <i>see also TLS-1.2</i> at 47-50, 57, 65, 72, 75, 92-93, 100. <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provide similar disclosures. <i>See SSL-3.0</i> at 25-27 (Server key exchange message includes <i>dh_p</i>, <i>dh_g</i> and <i>dh_Ys</i>), 28, 33, 39-41, 57, 61; <i>TLS-1.0</i> at 39-41 (Server key exchange message includes <i>dh_p</i>, <i>dh_g</i> and <i>dh_Ys</i>), 37-39, 43, 48, 52-53, 54-55, 71, 75; <i>TLS-1.1</i> at 42-44 (Server key exchange message includes <i>dh_p</i>, <i>dh_g</i> and <i>dh_Ys</i>), 40-41, 47, 53, 58-59, 61, 76; <i>see also</i> claim features [1a-1], [1c]-[1d], 6, 7. Because the grounds alleged here teaches the security engine is used to generate keys using the IPsec IKE or SSL/TLS protocol, it would need to receive the server key exchange message according to one of these protocols, which teaches using the security engine to receive Diffie-Hellman global values (e.g., <i>dh_p</i> and <i>dh_g</i>) and public Diffie-Hellman server key (e.g., <i>DH_Ys</i>). <i>See supra</i> [1a-1], [1c]-[1d]. Each IKE protocols (<i>RFC-5996</i> and <i>RFC-4306</i>) also teaches receiving Diffie-Hellman global values and a public Diffie-Hellman server key from the remote server. More specifically, <i>RFC-5996</i> discloses in the initial exchanges, “[t]he first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange [DH].” <i>RFC-5996</i> at 9; <i>see also RFC-5996</i> at 6, 9-16, 22, 33, 44-48, 52-53, 76, 80-83, 86-87, 121, 126, 128, 130-131, 133, 134, 137. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 7 (“The first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange [DH].”); <i>see also RFC-4306</i> at 4, 7-10, 21, 26-30, 33-34, 47, 50-52, 55-56, 88-89, 91-92, 95, 97. In addition, <i>RFC-5996</i> also discloses receiving Diffie-Hellman group and primes (the claimed Diffie-Hellman global values) from the remote server. <i>See RFC-5996</i> at 12 (“the Diffie-Hellman group that responder will select from [initiator’s] list of supported groups.”); <i>see also RFC-5996</i> at 12-16, 33, 45,

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	48, 76-77, 80-81, 83, 86-87, 121, 126, 128, 130-131, 133. <i>RFC-4306</i> provides similar disclosures. See <i>RFC-4306</i> at 10, 21, 27, 30, 47, 50-53, 55-56, 88, 91, 92, 95, 97. In addition, <i>SSL-3.0</i>, <i>TLS-1.0</i>, <i>TLS-1.1</i>, <i>TLS-1.2</i>, <i>RFC-5996</i> and <i>RFC-4306</i> all expressly incorporate by one or more Diffie-Hellman related references, such as <i>RFC2785</i>, <i>Hellman-770</i>, <i>Hellman-582</i>, or <i>Diffie</i> respectively, which all disclose receiving Diffie-Hellman global values and a public Diffie-Hellman server key from the remote server to calculate a shared key. See claims 6 and 7. Consistent with these teachings, the specification of the '838 patent describes Diffie-Hellman global values as, “e.g., a prime number and a primitive root of the prime number.” '838 pat. at 7:47-53. The grounds alleged here thus teach that “the security engine is further to receive Diffie-Hellman global values and a public Diffie-Hellman server key from the remote server.” The grounds alleged here include “stor[ing] at least one of the Diffie-Hellman global values and the public Diffie-Hellman server key in the secure memory.” A POSITA would have found it obvious to store at least one of the Diffie-Hellman global values and the public Diffie-Hellman server key in secure memory discussed in [1a-2]. To begin, <i>Bugiel</i> assumes the “availability of a secure hardware which provides a Trusted Execution Environment (TrEE) with . . . secure storage . . .” <i>Bugiel</i> at 222. <i>Bugiel</i> also teaches that its method relies on “the TrEE provided by secure hardware is tamper-protected.” <i>Bugiel</i> at 222, 233 (“credentials associated with the SSL/TLS channel (such as pre-master secret key and a session key) are protected by the TrEE”); see also <i>Bugiel</i> at 220-226, 228-235. Each of IKE and SSL/TLS protocols also emphasizes the importance of keeping Diffie-Hellman parameters secure for further computation of pre-master secret, master secret and encryption keys. See <i>TLS-1.2</i> at 92 (“Eavesdroppers who do not know the private values should not be able to find the Diffie-Hellman result (i.e., the pre_master_secret).”), 92-93 (explaining Diffie-Hellman parameters are “signed with a DSA or RSA certificate,” that “[t]emporary parameters [which would include the Diffie-Hellman global values and keys] are hashed with the hello.random values before signing to

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	ensure that attackers do not replay old parameters,” and that “[i]f the same DH keypair is to be used for multiple handshake . . . care must be taken to prevent small subgroup attacks”), 96 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications <i>must be secure</i>. In addition, the implementation must be free of security errors.”) (emphases added); <i>see also</i> <i>TLS-1.2</i> at 1, 4, 16, 34, 59, 91-97. <i>SSL-3.0</i>, <i>TLS-1.0</i> and <i>TLS-1.1</i> also provide similar disclosures. <i>See SSL-3.0</i> at 55-56 (“Eavesdroppers who do not know the private values should not be able to find the Diffie-Hellman result (i.e. the <code>pre_master_secret</code>)”), 57 (explaining Diffie-Hellman parameters are “signed with a DSA or RSA certificate,” that “[t]emporary parameters [which would include the Diffie-Hellman global values and keys] are hashed with the <code>hello.random</code> values before signing to ensure that attackers do not replay old parameters,” and that “[i]f the same DH keypair is to be used for multiple handshake . . . care must be taken to prevent small subgroup attacks”), 59 (“For SSL to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 9, 55-58, 17; <i>TLS-1.0</i> at 70 (“Eavesdroppers who do not know the private values should not be able to find the Diffie-Hellman result (i.e. the <code>pre_master_secret</code>)”), 71 (explaining Diffie-Hellman parameters are “signed with a DSA or RSA certificate,” that “[t]emporary parameters [which would include the Diffie-Hellman global values and keys] are hashed with the <code>hello.random</code> values before signing to ensure that attackers do not replay old parameters,” and that “[i]f the same DH keypair is to be used for multiple handshake . . . care must be taken to prevent small subgroup attacks”), 73 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 13, 29-30, 44-45, 69, 71-72; <i>TLS-1.1</i> at 75 (“Eavesdroppers who do not know the private values should not be able to find the Diffie-Hellman result (i.e. the <code>pre_master_secret</code>)”), 76 (explaining Diffie-Hellman parameters are “signed with a DSA or RSA certificate,” that “[t]emporary parameters [which would include the Diffie-Hellman global values and keys] are hashed with the <code>hello.random</code> values before signing to ensure that attackers do not replay old parameters,” and that “[i]f the same DH keypair is to be used for multiple handshake . . . care must be taken to prevent small subgroup attacks”), 80 (“For TLS to be able to provide a

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 14-15, 31, 48-49, 74-75, 77-80. Each IKE protocol also teaches storing the Diffie-Hellman values for use. <i>See RFC-5996</i> at 121 (“It is assumed that all Diffie-Hellman exponents are erased from memory after use”), 44-45 (reuse of Diffie-Hellman exponentials), 120-124 (Security Considerations), 130. <i>RFC-4306</i> also provides similar disclosures. <i>See RFC-4306</i> at 89 (“It is assumed that all Diffie-Hellman exponents are erased from memory after time.”), 27 (reuse of Diffie-Hellman Exponentials), 88-90 (Security Considerations). Because IKE, SSL/TLS protocols and <i>Bugiel</i> teach or suggest the Diffie-Hellman global values and the public Diffie-Hellman server key are essential security parameters for generating encryption keys in the IKE or SSL/TLS protocol, a POSITA would have found it obvious to store them in the secure memory discussed in [1a-2] to prevent attackers and for further computation of the secrets and encryption keys. Other prior art references also similarly describe storing key security parameters for secure processing in the secure memory. <i>See supra</i> [1a-2], [1f], [2]-[4] (describing storing key security parameters in the secure memory for the prior art references). As one example, <i>Chou</i> teaches that its “[s]ecurity processor 21 provides the various processing within HSM 20, including key authentication and the encrypting and/or decrypting operation to encrypt and/or decrypt data. . . . Secure memory 22 is utilized to store data that is processed and/or to be processed by security processor 21.” <i>Chou</i> ¶ [0021]; <i>see also id.</i> ¶ [0018] (“The term “secure” (such as secure domain, zone or boundary) is used herein to designate a zone controlled by HSM 20. Thus, components outside of the zone established by HSM 20 reside in a non-secure zone relative to HSM 20.”). <i>Chou</i> thus teaches secure memory 22 is kept within secure zone 80 for storing secure data that is processed and/or to be processed, including Diffie-Hellman global values and keys. In addition to <i>Chou</i>, other references also teach the secure memory only accessible to the security engine. <i>See supra</i> [1a-2] (each reference’s disclosure of secure memory).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Exemplary disclosures from the prior art include the following: <i>Bugiel</i> discloses the following at 222: Assumptions. We rely on availability of a secure hardware which provides a Trusted Execution Environment (TrEE) with following features: (i) isolated secure code execution, (ii) secure storage, (iii) integrity protection of secure execution environment. Such TrEEs can be provided by general purpose secure hardware such as M-Shield [8] and ARM TrustZone [9]. We assume the TrEE provided by secure hardware is tamper-protected. <i>See also Bugiel</i> at 220-226, 228-236. <i>See also TLS-1.2</i> at 1, 4, 16, 34, 47-53, 57, 59, 65, 72, 75, 91-97, 100. <i>See also SSL-3.0</i> at 1, 9, 13, 17, 25-30, 33, 39-41, 44-45, 55-59, 61, 69, 71-72. <i>See also TLS-1.0</i> at 1, 4, 13-15, 29-31, 37-41, 43-45, 48-49, 52-53, 54-55, 69-75, 77-80. <i>See also TLS-1.1</i> at 1, 4, 14-15, 31, 40-44, 47-49, 53, 58-59, 61, 74-80. <i>See also RFC-5996</i> at 6, 9-16, 22, 33, 44-48, 52-53, 76-77, 80-83, 86-87, 120-124, 126, 128, 130-131, 133-134, 137. <i>See also RFC-4306</i> at 4, 7-10, 21, 26-30, 33-34, 47, 50-53, 55-56, 88-92, 95, 97. <i>See also claims</i> 1, 2, 4-7.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
Claim 9	
[9pre] A method comprising:	To the extent the preamble is limiting, the grounds alleged at the beginning of this chart disclose or suggest “[a] method.” The grounds alleged here implement <i>Bugiel</i>’s communications method in <i>Bugiel</i>’s secure execution environment, including to implement IPsec IKE or SSL/TLS Handshake Protocol messages, to establish a secure communication. For example, <i>Bugiel</i> teaches a method by disclosing performing secure communication using known protocols, such as SSL/TLS or IPsec, etc. <i>See Bugiel</i> at 223 (“letting the trusted and untrusted code cooperate in establishing a secure channel between the user platform and the web server.”), 222 (“our solution supports this method”); <i>see general Bugiel</i> 219-236 (the TruWalletM method implemented on commodity secure hardware providing secure execution environment, e.g., M-Shield); <i>see supra</i> [1]. In addition, <i>Chou</i> discloses “method for providing hardware security.” <i>Chou</i> at Title; <i>see also Chou</i> ¶ [0001], ¶ [0010], claims 16-20. In addition, <i>Tkacik</i> discloses “[a] method of operating an encryption apparatus” <i>Tkacik</i> at claims 10-20. Further, a POSITA would have understood that the processes described in Figures 5-8 are methods. <i>See Tkacik</i> Figures 5-8. In addition, <i>Paaske</i> also discloses methods of using a TPM for cryptographic processing. <i>See generally Paaske</i>, including e.g., <i>Paaske</i> ¶ [0007], ¶ [0009], ¶¶ [0021]-[0022], claims 1-13, 25-39. In addition, <i>Tugenberg</i> explains that “Secure Socket Layer (SSL), for example, is a standard technique used in the internet community to establish [a secure] session.” <i>Tugenberg</i> at 5:47-54, title (method); <i>see also Tugenberg</i> at Abstract, 1:37-55, 2:3-21, 4:54-58, 6:12-20, 8:1-20, claims 1-19. In addition, <i>Weise</i> discloses “methods.” <i>Weise</i> ¶ [0021], ¶ [0043]. In addition, <i>Buer754</i> discloses “Key management system and method.” <i>Buer754</i> at Title, claims 1 (“A method of cryptographically processing data in a host cryptographic system”), claim 16 (“A method

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	for managing cipher key of a host cryptographic system”), Abstract, 2:17-60, 10:23-28, 10:57-61, 15:46-60, 16:20-26, claims 1-8, 12, 15-20. In addition, <i>Gibod</i> discloses “secure processing systems and methods.” <i>Gibod</i> at Title, Abstract, ¶ [0001], ¶¶ [0003]-[0004], ¶¶ [0012]-[0014], ¶¶ [0016]-[0017], ¶ [0019], ¶ [0033], ¶ [0045], ¶¶ [0047]-[0049], ¶¶ [0053]-[0056], ¶ [0058], ¶¶ [0060]-[0063], ¶ [0065], ¶¶ [0069]-[0071], Figs. 2-4, claims 1-19. In addition, <i>Aaraj</i> discloses “the methodology is applicable to any extensible processor platform.” <i>Aaraj</i> at 8:20-8:21; <i>see also Aaraj</i> at 8:9, 8:15-8:16, 8:30-8:31. In addition, <i>Mujtaba</i> discloses “the processes or methods.” <i>Mujtaba</i> ¶¶ [0066]-[0068], claims 12-24 <i>In addition, TLS-1.2</i> discloses at “The Transport Layer Security (TLS) Protocol.” <i>TLS-1.2</i> at 1; 6 (“TLS seeks to provide a framework into which new public key and bulk encryption methods can be incorporated as necessary”); <i>SSL-3.0</i> at 1 (“The SSL Protocol”), 4 (“SSL seeks to provide a framework into which new public key and bulk encryption methods can be incorporated as necessary”); <i>TLS-1.0</i> at 1 (“The TLS Protocol”), 4 (“TLS seeks to provide a framework into which new public key and bulk encryption methods can be incorporated as necessary”); <i>TLS-1.1</i> at 1 (“The Transport Layer Security (TLS) Protocol”), 4 (“TLS seeks to provide a framework into which new public key and bulk encryption methods can be incorporated as necessary”); <i>RFC-5996</i> at title (“Internet Key Exchange Protocol Version 2 (IKEv2)”), 1 (“IKE is a component of IPsec used for performing mutual authentication and establishing and maintaining security associations (SAs).”); <i>RFC-4306</i> at title (“Internet Key Exchange (IKEv2) Protocol”), 1 (“IKE is a component of IPsec used for performing mutual authentication and establishing and maintaining security associations (SAs).”); <i>see also supra</i> claim features [1b]-[1g], [2]-[8]. Exemplary disclosures of the prior art include the following: <i>Chou</i> is titled:

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Apparatus and method for providing hardware security <i>Bugiel</i> discloses the following at 223: [O]ur design relies on isolated execution of trusted code within the TrEE. However, currently TrEEs provided by commodity secure hardware are very resource constrained, e.g., M-Shield has about 10-20 Kb memory available in the secure mode. On the other hand, a standard user/password authentication protocol assumes transmission of the password to the server through a SSL/TLS connection. In our design this implies running the code handling SSL/TLS communication within the TrEE in order to (i) protect user credentials and (ii) be compatible with legacy web-servers. However, this is not possible due to size of this code. Thus, one of the main challenges we face is to handle SSL/TLS channels with a web server in a secure way as this cannot be done within the TrEE. We overcome these limitations by letting the trusted and untrusted code cooperate in establishing a secure channel between the user platform and the web server. <i>See general Bugiel</i> 219-236. <i>See also Chou</i> ¶ [0001], ¶ [0010], claims 16-20, Title. <i>See also Tkacik</i> Figures 5-8 and related discussion, claims 10-20. <i>See generally Paaske</i>, including e.g., <i>Paaske</i> ¶ [0007], ¶ [0009], ¶¶ [0021]-[0022], claims 1-13, 25-39. <i>See also Tugenberg</i> at Title, Abstract, 1:37-55, 2:3-21, 4:54-58, 5:47-54, 6:12-20, 8:1-20, claims 1-19. <i>See also Buer754</i> at Title, claims 1-8, 12, 15-20, Abstract, 2:17-60, 10:23-28, 10:57-61, 15:46-60, 16:20-26.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also Gibod</i> at Title, Abstract, ¶ [0001], ¶¶ [0003]-[0004], ¶¶ [0012]-[0014], ¶¶ [0016]-[0017], ¶ [0019], ¶ [0033], ¶ [0045], ¶¶ [0047]-[0049], ¶¶ [0053]-[0056], ¶ [0058], ¶¶ [0060]-[0063], ¶ [0065], ¶¶ [0069]-[0071], Figs. 2-4, claims 1-19. <i>See also Aaraj</i> at 8:9, 8:15-8:16, 8:20-8:21, 8:30-8:31. <i>See also Weise</i> ¶ [0021], ¶ [0043]. <i>See also Mujtaba</i> ¶ [0066]-[0068], claims 12-24. <i>See generally TLS-1.2</i>, including e.g., Title, 1, 6. <i>See generally SSL-3.0</i>, including, e.g. Title, 1, 4. <i>See generally TLS-1.0</i>, including e.g., Title, 1, 4. <i>See generally TLS-1.1</i>, including e.g., Title, 1, 4. <i>See generally RFC-5996</i>, including e.g., Title, 1. <i>See generally RFC-4306</i> at Title, 1. <i>See also supra</i> claims 1-8.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
[9a] generating a random nonce in a security engine that is separate from a processor core of a system-on-a-chip of a client device;	The grounds alleged at the beginning of this chart disclose or suggest “generating a random nonce in a security engine that is separate from a processor core of a system-on-a-chip of a client device.” <i>See supra</i> [1pre] (“A system-on-a-chip apparatus”), [1a-1] (“a system-on-a-chip comprising a security engine that is separate from a processor core of the system-on-a-chip”), [1b] (“generate a random nonce for initiating a request for a secure communication session with a remote server over a network using the nonce”); <i>see also</i> [1].
[9b] initiating, using the client device, a request for a secure communication session with a remote server over a network, the request including the random nonce;	The grounds alleged at the beginning of this chart disclose or suggest “initiating, using the client device, a request for a secure communication session with a remote server over a network, the request including the random nonce.” <i>See supra</i> [1b] (“generate a random nonce for initiating a request for a secure communication session with a remote server over a network using the nonce”); <i>see also</i> [1].
[9c] performing a cryptographic key exchange, using the security engine of the system-on-a-chip, with the remote server;	The grounds alleged at the beginning of this chart disclose or suggest “performing a cryptographic key exchange, using the security engine of the system-on-a-chip, with the remote server.” <i>See supra</i> [1c] (“perform a cryptographic key exchange with the remote server”); <i>see also</i> [1].
[9d] generate a symmetric session key to encrypt messages sent to the remote server and decrypt messages received from the remote server during	The grounds alleged at the beginning of this chart disclose or suggest to “generate a symmetric session key to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session.” <i>See supra</i> [1d] (“generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session”); <i>see also</i> [1].

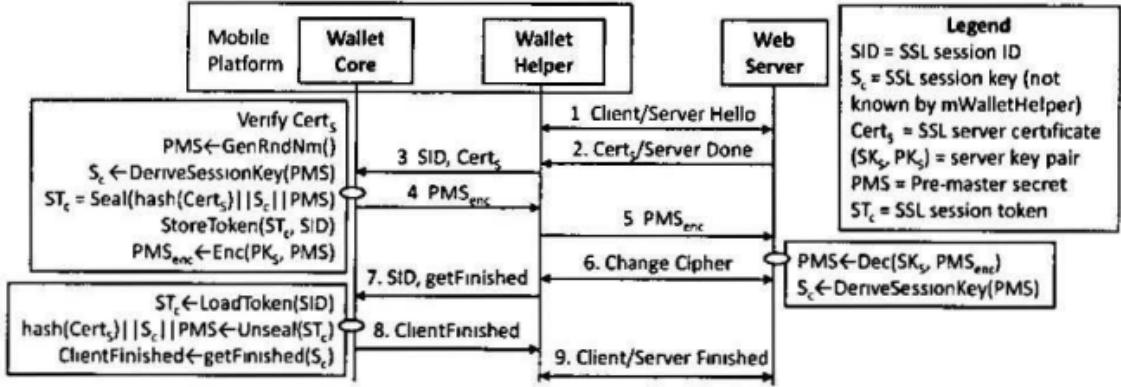
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
the secure communication session;	
[9e] encrypting the session key based on a security key that was encoded in a secure memory of the security engine during a manufacturing process of the system-on-a-chip;	The grounds alleged at the beginning of this chart disclose or suggest “encrypting the session key based on a security key that was encoded in a secure memory of the security engine during a manufacturing process of the system-on-a-chip.” <i>See supra</i> [1a-2] (“has a secure memory accessible only by the security engine”), [1a-3] (“wherein the secure memory includes a security key that was encoded in the secure memory during a manufacturing process of the system-on-a-chip”), [1e] (“encrypt the symmetric session key based on the security key”); <i>see also</i> [1].
[9f] storing the encrypted session key in the secure memory of the security engine of the system-on-a-chip; and	The grounds alleged at the beginning of this chart disclose or suggest “storing the encrypted session key in the secure memory of the security engine of the system-on-a-chip.” <i>See supra</i> [1a-2] (“has a secure memory accessible only by the security engine”), [1f] (“store the encrypted session key in the secure memory”); <i>see also</i> [1].
[9g] establishing, using the client device, the secure communication session with the remote server using the session key.	The grounds alleged at the beginning of this chart disclose or suggest “establishing, using the client device, the secure communication session with the remote server using the session key.” <i>See supra</i> [1g] (“the system-on-a-chip to establish the secure communication session with the remote server over the network using the session key.”); <i>see also</i> [1].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
Claim 10	
[10a] The method of claim 9, wherein initiating a request for a secure communication session comprises sending a ClientHello message, including the nonce, to the remote server; and	The grounds alleged at the beginning of this chart disclose or suggest “initiating a request for a secure communication session comprises sending a ClientHello message, including the nonce, to the remote server.” The grounds alleged here implements <i>Bugiel</i>’s communications method in <i>Bugiel</i>’s secure execution environment, including to implement IPsec IKE or SSL/TLS Handshake Protocol messages, to establish a secure communication. <i>Bugiel</i> discloses that “The client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters” as part of standard secure communication protocols, e.g., SSL/TLS or IPsec IKE. <i>Bugiel</i> at 225-226, Figs. 2-3. In addition, at the time of the '838 patent, it was well-known to a POSITA to send a ClientHello message, including the nonce, to the remote server to initiate a secure communication session. For example, <i>Chou</i> teaches that PKA module 56 (part of the claimed “security engine”) performs cryptographic key exchange algorithms like Diffie-Hellman shared key computation and RSA key gen. <i>See Chou</i> ¶¶ [0043]-[0053], ¶ [0034] (RNG), ¶¶ [0041]-[0042], claim 11, Fig. 3. In addition to <i>Chou</i>, other prior art references also teach this aspect. <i>See supra</i> [1b], [2], [9b]. The grounds alleged here implement known protocols for secure communication (such as IPsec IKE or SSL/TLS Handshake Protocol messages, e.g., TLS version 1.2, etc.) as <i>Bugiel</i> teaches, including the teaching in IPsec IKE or SSL/TLS to generate a ClientHello.random (a random nonce) and send a ClientHello message, including the ClientHello.random random nonce, to a remote server to initiate a secure TLS connection. <i>See Bugiel</i> at 225-227; <i>see supra</i> [1], [9].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	 Fig. 2. SSL/TLS handshake In addition, Figures 1 and 2 of <i>TLS-1.2</i> illustrate, the first step of the TLS handshake is to send a ClientHello message. <i>See TLS-1.2</i> at 34-37; <i>see also TLS-1.2</i> at 38-45, 47, 59-60, 62, 64, 70, 71, 75, 86-90, 92, 94. <i>TLS-1.2</i> further teaches to include a random nonce called ClientHello.random (the claimed “nonce”) in the ClientHello message. <i>See TLS-1.2</i> at 34 (“The client sends a ClientHello message [T]wo random values are generated and exchanged: ClientHello.random”); <i>see also TLS-1.2</i> at 4, 6, 17-18, 26, 33-43, 52, 64, 71, 73, 77, 85, 90, 92, 94, 100.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	 <pre> sequenceDiagram participant Client participant Server Client->>Server: ClientHello Server->>Client: ServerHello Server->>Client: [ChangeCipherSpec] Server->>Client: Finished Client->>Server: [ChangeCipherSpec] Client->>Server: Finished Client->>Server: Application Data Server->>Client: Application Data </pre> Figure 2. Message flow for an abbreviated handshake <i>SSL-3.0, TLS-1.0, TLS-1.1</i> also provide similar disclosures. See <i>TLS-1.1</i> at 32 (“two random values are generated and exchanged: ClientHello.random and ServerHello.random”), 33-34, 36, 5, 16, 24, 31, 36-39, 44, 52-53, 57-58, 64, 70, 73, 75-76, 78, 83; <i>TLS-1.0</i> at 30 (“two random values are generated and exchanged: ClientHello.random and ServerHello.random”), 31-32, 34, 4, 14-15, 21-23, 29, 35-37, 41, 47, 51-52, 56, 64, 67-68, 70-72; <i>SSL-3.0</i> at 18 (“two random values are generated and exchanged: ClientHello.random and ServerHello.random”), 19-21, 4, 11, 22-24, 26, 32, 34-35, 38, 49, 52-53, 56-59; see <i>supra</i> [1b]. At the time of the '838 patent, it was common knowledge that ClientHello.random and ServerHello.random are random nonces exchanged between client and remote server to avoid “replay” attack in SSL/TLS protocols (see <i>TLS-1.2</i> at 91-92; <i>SSL-3.0</i> at 55, 57; <i>TLS-1.0</i> at 69-71; <i>TLS-1.1</i> at 74-76).²¹ In addition, <i>TLS-1.0</i> incorporates by reference the original SSL patent <i>Elgamal</i>. See <i>TLS-1.0</i> at

²¹ Before the earliest possible priority date of the '838 patent, many patent documents describe the random numbers exchanged between the server and client as random nonce. See U.S. Patent No. 6,643,774 to McGarvey (e.g., at 7:16-8:3), which was filed on April 8, 1999 and patented on November 4, 2003; U.S. Patent No. 7,039,946 to Binding et al. (e.g., at 10:41-57, 11:52-12:4, 14:52-15:6), which was filed on October 12, 1999 and patented on May 2, 2006; U.S. Patent No. 6,775,772 to Binding et al. (e.g., at 9:64-10:13, 11:5-23, 14:1-21), which was filed on October 12, 1999 and patented on August 10, 2004; U.S. Patent No. 6,694,431 to Binding et al. (e.g., at 10:10-26, 11:18-37, 14:15-35), which was filed on October 12, 1999 and patented on February 17, 2004; U.S. Patent No. 6,751,731 to Binding

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

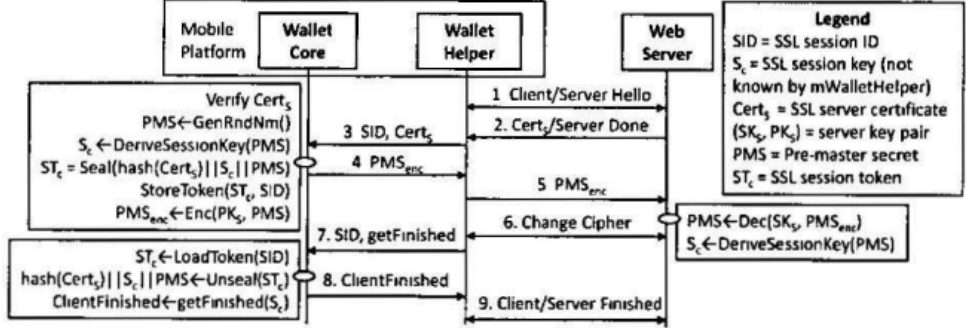
The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	74. <i>Elgamal</i> also teaches including “a random number used to ensure channel integrity” in the client-hello message, which the server responds with a server-hello message including a server random number. <i>Elgamal</i> at 7:13-40; <i>see also id.</i> at Figs. 4-7. <i>Elgamal</i> further describes the random number as “nonce” because they are “used to defeat ‘playback’ attacks.” <i>Elgamal</i> at 33:24-31. In addition, each IKE protocol (<i>RFC-5996</i> or <i>RFC-4306</i>) also teaches initiating a request for a secure communication session comprises sending a ClientHello message, including the nonce, to the remote server. For example, <i>RFC-5996</i> teaches the following at 10: <pre> The initial exchanges are as follows: Initiator Responder ----- HDR, SAi1, KEi, Ni --> HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. The SAi1 payload states the cryptographic algorithms the initiator supports for the IKE SA. The KE payload sends the initiator's Diffie-Hellman value. Ni is the initiator's nonce. <-- HDR, SAR1, KEr, Nr, [CERTREQ]</pre>

et al. (e.g., 9:63-10:12, 11:12-18, 14:15-35), which was filed on October 12, 1999 and patented on June 15, 2004; U.S. Patent Application Publication No. 2010/0205443 to Zhao et al. (e.g., at ¶ [0114], ¶¶ [0175]-[0177], ¶ [0186]), which was filed on April 23, 2010 and published on August 12, 2010; U.S. Patent Application Publication No. 2011/0231653 to Bollay et al., (e.g., at ¶¶ [0003]-[0005], ¶¶ [0022]-[0029], ¶ [0036], ¶ [0062], ¶ [0070], ¶ [0090], ¶ [0137]), which was filed on December 13, 2010 and published on September 22, 2011; U.S. Patent No. 5,657,390 to Elgamal et al (e.g., at 6:35-55, 33:24-37, Figs. 9-11), which was filed on August 25, 1995 and patented on August 12, 1997; U.S. Patent Application Publication No. 2010/0042824 to Lee et al. (e.g., at ¶ [0074], ¶ [0081], ¶¶ [0087]-[0088], ¶ [0118], ¶ [0130]), which was filed on August 14, 2009 and published on February 18, 2010; *Buer-754* at 1:36-41, 1:61-2:8, 6:13-31, 13:21-37, 13:45-65, 14:11-34, 14:43-55; EP1706956B1 to Cam-Winget et al. (e.g., at ¶¶ [0139]-[0142]), which was filed on January 10, 2005 and published on October , 2006; U.S. Patent Application Publication No. 2004/0005061 to Buer et al. (e.g., at ¶ [0005], ¶¶ [0009]-[0010], ¶¶ [0050]-[0051], ¶¶ [0108]-[0115], ¶¶ [0117]-[0124], ¶¶ [0127]-[0134], ¶¶ [0136]-[0140]), which was filed on July 8, 2002, and published on January 8, 2004.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>RFC-5996</i> at 10. Here, the initiator initiates a request for a secure communication session comprising sending a ClientHello message, including the nonce <i>N_i</i>, to the remote server, i.e., the responder. <i>RFC-5996</i> at 10; <i>see also RFC-5996</i> at 6, 9-11, 13-16, 33, 37-39, 44, 46-48, 50, 52-53, 62, 69, 74, 96, 122, 134, 137. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 4 (“The first request/response of an IKE session (IKE_SA_INIT) negotiates security parameters for the IKE_SA, sends nonces, and sends Diffie-Hellman values.”), 7-8, 10, 19, 23, 26, 28-30, 32-34, 39, 45, 64, 69; <i>see also</i> [1b]. Exemplary disclosures include the following: <i>Bugiel</i> discloses the following at 225-227 (4.1 Establishing SSL/TLS connection and managing two logical SSL/TLS subchannels): Figure 2 illustrates the SSL handshake protocol. The client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters. In the following steps the server's certificate <i>Certs</i> is sent and verified, the SSL/TLS session key <i>Sc</i> is generated (which is never disclosed to WalletHelper) and sealed (together with pre-master secret <i>PMS</i> and the hash value <i>hash(Certs)</i>). The resulting session-token <i>STc</i> is stored in a secure storage and can be loaded and unsealed later using the session identifier <i>SID</i>. ... All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	 Fig. 2. SSL/TLS handshake See also <i>Bugiel</i> at 235-236. <i>TLS-1.2</i> discloses the following at 34: The TLS Handshake Protocol involves the following steps: - Exchange hello messages to agree on algorithms, exchange random values, and check for session resumption. ... These goals are achieved by the handshake protocol, which can be summarized as follows: The client sends a client hello message to which the server must respond with a server hello message, or else a fatal error will occur and the connection will fail. The client hello and server hello are used to establish security enhancement capabilities between client and server. The client hello and server hello establish the following attributes: Protocol Version, Session ID, Cipher Suite, and Compression Method. Additionally, two random values are generated and exchanged: ClientHello.random and ServerHello.random. See also <i>TLS-1.2</i> at 4, 6, 17-18, 26, 33-45, 47, 52, 59-60, 62, 64, 70-71, 73, 75, 77, 85-92, 94, 100.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also TLS-1.1</i> at 5, 16, 24, 31-34, 36-39, 44, 52-53, 57-58, 64, 70, 73-76, 78, 83. <i>See also TLS-1.0</i> at 4, 14-15, 21-23, 29-32, 34-37, 41, 47, 51-52, 56, 64, 67-72, 74. <i>See also SSL-3.0</i> at 4, 11, 18-24, 26, 32, 34-35, 38, 49, 52-53, 55-59. <i>See also RFC-5996</i> at 6, 9-11, 13-16, 33, 37-39, 44, 46-48, 50, 52-53, 62, 69, 74, 96, 122, 134, 137. <i>See also RFC-4306</i> at 4, 7-8, 10, 19, 23, 26, 28-30, 32-34, 39, 45, 64, 69. <i>See also supra</i> [1b], [2]; <i>see also</i> [1], [9].
[10b] further comprising: receiving a ServerHello message from the remote server, the ServerHello message including a server nonce; and storing the server nonce in the secure memory of the security engine of the system-on-a-chip.	The grounds alleged at the beginning of this chart disclose or suggest “receiving a ServerHello message from the remote server, the ServerHello message including a server nonce; and storing the server nonce in the secure memory of the security engine of the system-on-a-chip.” The grounds alleged here use the security engine and secure memory discussed in [1a-1] and [1a-2] to implement <i>Bugiel</i>’s communications method in <i>Bugiel</i>’s secure execution environment, including to implement IPsec IKE or SSL/TLS Handshake Protocol messages, to establish a secure communication. The alleged combinations include the teaching in IPsec IKE or SSL/TLS Handshake Protocol to receive a ServerHello message from the remote server, including a server nonce, and to store the server nonce in the secure memory of the security engine of the system-on-a-chip discussed in [1]-[2], [9]. During the SSL/TLS Handshake Protocol, the first step is for a client and server to exchange ClientHello (the claimed “request for the secure communication session”) and ServerHello messages (the claimed “a response message to the request”), wherein the ServerHello message includes a ServerHello.random (the claimed “server nonce”). For example, <i>TLS-1.2</i> illustrates the following in Figure 2 below; <i>see also TLS-1.2</i> at 42-43 (“The server will send this message [ServerHello message]

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	in response to a client hello message”), 38-47, 17-18, 20, 26, 33-38, 52, 55, 64, 70-71, 73, 75, 77, 85, 87-88, 92, 94, 100. <pre> sequenceDiagram participant Client participant Server Note over Client: ClientHello Client->>Server: Note over Server: ServerHello Note over Server: [ChangeCipherSpec] Note over Server: Finished Server-->>Client: Note over Client: [ChangeCipherSpec] Note over Client: Finished Note over Client: Application Data Client->>Server: Note over Server: Application Data </pre> Figure 2. Message flow for an abbreviated handshake <i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provides similar disclosures as <i>TLS-1.2</i>. See <i>SSL-3.0</i> at 18 (“These processes are performed in the handshake protocol, which can be summarized as follows: The client sends a client hello message to which the server must respond with a server hello message, or else a fatal error will occur and the connection will fail. . . . Additionally, two random values are generated and exchanged: ClientHello.random and ServerHello.random.”), 24 (“The server processes the client hello message and responds with . . . server hello message.”); see also <i>SSL-3.0</i> at 18-28, 32-35, 37-38, 41, 49, 51, 56-59; <i>TLS-1.0</i> at 29-30 (“The TLS Handshake Protocol involves the following steps: Exchange hello messages . . . exchange random values, . . . These goals are achieved by the handshake protocol, which can be summarized as follows: The client sends a client hello message to which the server must respond with a server hello message, or else a fatal error will occur and the connection will fail. . . . Additionally, two random values are generated and exchanged: ClientHello.random and ServerHello.random”), 36 (“The server will send this message [ServerHello message] in response to a client hello message”), Figs. 1-2; see also <i>TLS-1.0</i> at 14-15, 21-23, 29-42, 47, 51-52, 54, 56, 64, 66, 70-72; <i>TLS-1.1</i> at 31-32 (“The TLS Handshake Protocol involves the following steps: Exchange hello messages . . . exchange random values, . . . These goals are achieved by the handshake protocol, which can be summarized as follows: The client sends a client hello message to which the server must respond with a server hello message, or else a fatal error will occur and the connection will fail. . . . Additionally, two random values are generated and exchanged: ClientHello.random and ServerHello.random”), 39

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	(“The server will send this message [ServerHello message] in response to a client hello message”), Figs. 1-2; <i>see also TLS-1.1</i> at 16, 24, 31-46, 52-53, 57-58, 60, 64, 70-71, 76, 78, 83. As discussed in [1b], at the time of the '838 patent, it was common knowledge that ClientHello.random and ServerHello.random are random nonces exchanged between client and remote server to avoid “replay” attack in SSL/TLS protocols (<i>see TLS-1.2</i> at 91-92; <i>SSL-3.0</i> at 55, 57; <i>TLS-1.0</i> at 69-71; <i>TLS-1.1</i> at 74-76).²² In addition, <i>TLS-1.0</i> incorporates by reference the original SSL patent <i>Elgamal</i>. <i>See TLS-1.0</i> at 74. <i>Elgamal</i> also teaches including “a random number used to ensure channel integrity” in the client-hello message, which the server responds with a server-hello message including a server random number. <i>Elgamal</i> at 7:13-40; <i>see also id.</i> at Figs. 4-7. <i>Elgamal</i> further describes the random number as “nonce” because they are “used to defeat ‘playback’ attacks.” <i>Elgamal</i> at 33:24-31. <i>See, e.g., Callaway Golf Co. v. Acushnet Co.</i>, 576 F.3d 1331, 1346 (Fed. Cir. 2009) (“To incorporate

²² Before the earliest possible priority date of the '838 patent, many patent documents describe the random numbers exchanged between the server and client as random nonce. *See* U.S. Patent No. 6,643,774 to McGarvey (e.g., at 7:16-8:3), which was filed on April 8, 1999 and patented on November 4, 2003; U.S. Patent No. 7,039,946 to Binding et al. (e.g., at 10:41-57, 11:52-12:4, 14:52-15:6), which was filed on October 12, 1999 and patented on May 2, 2006; U.S. Patent No. 6,775,772 to Binding et al. (e.g., at 9:64-10:13, 11:5-23, 14:1-21), which was filed on October 12, 1999 and patented on August 10, 2004; U.S. Patent No. 6,694,431 to Binding et al. (e.g., at 10:10-26, 11:18-37, 14:15-35), which was filed on October 12, 1999 and patented on February 17, 2004; U.S. Patent No. 6,751,731 to Binding et al. (e.g., 9:63-10:12, 11:12-18, 14:15-35), which was filed on October 12, 1999 and patented on June 15, 2004; U.S. Patent Application Publication No. 2010/0205443 to Zhao et al. (e.g., at ¶ [0114], ¶¶ [0175]-[0177], ¶ [0186]), which was filed on April 23, 2010 and published on August 12, 2010; U.S. Patent Application Publication No. 2011/0231653 to Bollay et al., (e.g., at ¶¶ [0003]-[0005], ¶¶ [0022]-[0029], ¶ [0036], ¶ [0062], ¶ [0070], ¶ [0090], ¶ [0137]), which was filed on December 13, 2010 and published on September 22, 2011; U.S. Patent No. 5,657,390 to Elgamal et al (e.g., at 6:35-55, 33:24-37, Figs. 9-11), which was filed on August 25, 1995 and patented on August 12, 1997; U.S. Patent Application Publication No. 2010/0042824 to Lee et al. (e.g., at ¶ [0074], ¶ [0081], ¶¶ [0087]-[0088], ¶ [0118], ¶ [0130]), which was filed on August 14, 2009 and published on February 18, 2010; *Buer-754* at 1:36-41, 1:61-2:8, 6:13-31, 13:21-37, 13:45-65, 14:11-34, 14:43-55; EP1706956B1 to Cam-Winget et al. (e.g., at ¶¶ [0139]-[0142]), which was filed on January 10, 2005 and published on October , 2006; U.S. Patent Application Publication No. 2004/0005061 to Buer et al. (e.g., at ¶ [0005], ¶¶ [0009]-[0010], ¶¶ [0050]-[0051], ¶¶ [0108]-[0115], ¶¶ [0117]-[0124], ¶¶ [0127]-[0134], ¶¶ [0136]-[0140]), which was filed on July 8, 2002 and published on January 8, 2004.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	matter by reference, a host document must contain language ‘clearly identifying the subject matter which is incorporated and where it is to be found.’” (quoting <i>In re De Seversky</i>, 474 F.2d 671, 674 (CCPA 1973))) Additionally, each SSL/TLS protocol explains the variable server_random (i.e., the server random nonce) is part of the SecurityParameters data structure. <i>See TLS-1.2</i> at 18, 77; <i>see also SSL-3.0</i> at 21-25; <i>TLS-1.0</i> at 15, 56; <i>TLS-1.1</i> at 16, 64. According to <i>TLS-1.2</i>, this server nonce is needed to generate the master key, encryption key_block, etc. <i>See TLS-1.2</i> at 26, 34, 64, 85, 92, 94; <i>see also SSL-3.0</i> at 26, 32-35, 49, 58-59; <i>TLS-1.0</i> at 21-23, 29, 41, 47, 72; <i>TLS-1.1</i> at 24, 31, 44, 52-53, 78; <i>see supra</i> claim features [1c]-[1d]. <i>TLS-1.2</i> also explains: “For TLS to be able to provide a secure connection, . . . keys . . . must be secure.” <i>TLS-1.2</i> at 96; <i>see also TLS-1.2</i> at 1, 4, 16, 34, 59, 85, 91-97. <i>SSL-3.0</i>, <i>TLS-1.0</i> and <i>TLS-1.1</i> also provide similar disclosures. <i>See SSL-3.0</i> at 59 (“For SSL to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 9, 49-50, 55-58, 17; <i>TLS-1.0</i> at 73 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 13, 29-30, 64-65, 69, 71-72; <i>TLS-1.1</i> at 80 (“For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.”), 1, 4, 14-15, 31, 70, 74-75, 77-80. Knowing the random nonces, an attacker would be able to derive the keys based on the random nonces, thereby making the keys unsecure. Each SSL/TLS protocol is based on the premise that “attackers have substantial computational resources and cannot obtain secret information from sources outside the protocol.” <i>TLS-1.2</i> at 91 (Security Analysis); <i>see also SSL-3.0</i> at 55; <i>TLS-1.0</i> at 69; <i>TLS-1.1</i> at 74. Keeping the random nonce in secure memory helps ensure this core assumption remains true and that attackers cannot obtain secret information, such as the random nonce, by hacking into insecure memory.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

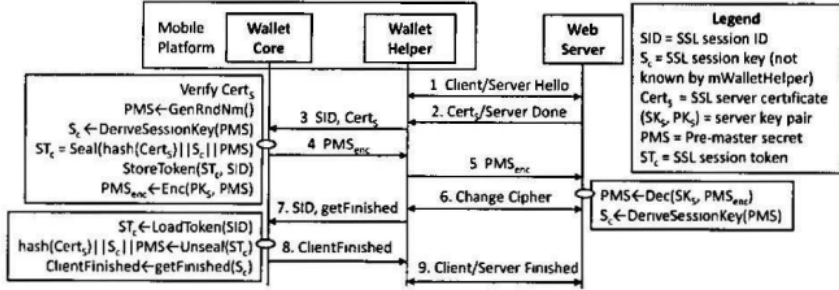
The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Likewise, <i>Bugiel</i> explains that its method assumes that “availability of a secure hardware which provides a Trusted Execution Environment (TrEE) with . . . secure storage” <i>Bugiel</i> at 222. <i>Bugiel</i> also teaches that its method relies on “the TrEE provided by secure hardware is tamper-protected.” <i>Bugiel</i> at 222, 233 (“credentials associated with the SSL/TLS channel (such as pre-master secret key and a session key) are protected by the TrEE”); <i>see also Bugiel</i> at 220-226, 228-235. In the alleged combination, each one of <i>Bugiel</i> in combination with one of <i>Chou</i>, <i>Tkacik</i>, <i>Paaske</i>, <i>Tugenberg</i>, <i>Aaraj</i>, <i>Gibod</i>, <i>Mujtaba</i>, or <i>Buer754</i> provides such hardware as discussed in [1a-1] and [1a-2]. As one example, as <i>Chou</i>’s Figure 1 illustrates, for example, <i>Chou</i>’s “[s]ecure memory 22 is utilized to store data that is processed and/or to be processed by security processor 21.” <i>Chou</i> ¶ [0021]. <i>Chou</i> also explains that a secure zone is established by and for the components of HSM 20, whereas components 11-14, for example, reside outside of that zone and are regarded as insecure. <i>See id.</i> ¶¶ [0020]-[0021]. <i>Chou</i> thus teaches that secure memory 22 securely stores data—including the server nonce—that is processed and/or to be processed by the security engine. In addition to <i>Chou</i>, other references also disclose using secure memory to keep data secure. <i>See supra</i> [1a-2], [1f], [2], [4], [8] for additional discussion of each reference’s disclosure of storing security parameters used in the secure communication protocol in the secure memory of the security engine. To operate on the server nonce securely when calculating other shared secrets and keys, a POSITA would have found it obvious to store the server random nonce in a secure memory to protect it from attackers and to derive various keys securely. <i>See TLS-1.2</i> at 25-26 (Key Calculation), 64 (Computing the Master Secret); <i>see also TLS-1.2</i> at 34, 94; <i>see also SSL-3.0</i> at 26, 32-35, 58-59; <i>TLS-1.0</i> at 21-23, 29, 41, 47, 72; <i>TLS-1.1</i> at 24, 31, 44, 52-53, 78; <i>see supra</i> claim features [1c]-[1d]. Additionally, each IKE protocol (<i>RFC-5996</i> or <i>RFC-4306</i>) also teaches “receiving a ServerHello message from the remote server, the ServerHello message including a server nonce; and storing the server nonce in the secure memory of the security engine of the system-on-a-chip.” For example, <i>RFC-5996</i> discloses in response to the initiator’s message, the responder “sends its nonce in the Nr payload” (the claimed “server nonce”) in both the initial exchange and

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>						
	CREATE_CHILD_SA exchange. <i>RFC-5996</i> at 10-11; <i>see also RFC-5996</i> at 6, 9-11, 13-16, 31, 33, 37-39, 44, 46-48, 50, 52-53, 62, 69, 74, 96, 121-122, 134, 137. The initial exchanges are as follows: <table style="border: none; width: 100%;"> <tr> <td style="width: 50%; text-align: left;">Initiator</td> <td style="width: 50%; text-align: right;">Responder</td> </tr> <tr> <td colspan="2" style="text-align: center;">-----</td> </tr> <tr> <td colspan="2">HDR, SAi1, KEi, Ni --></td> </tr> </table> HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. The SAi1 payload states the cryptographic algorithms the initiator supports for the IKE SA. The KE payload sends the initiator's Diffie-Hellman value. Ni is the initiator's nonce. <-- HDR, SAR1, KEr, Nr, [CERTREQ] <i>RFC-5996</i> at 10. <i>RFC-4306</i> provides similar disclosures. <i>See RFC-4306</i> at 8 (“The responder chooses a cryptographic suite from the initiator’s offered choices and expresses that choice in the SAR1 payload, completes the Diffie-Hellman exchange with the Ker payload, and sends its nonce in the Nr payload.”), 4, 7, 10, 19, 23, 26, 28-30, 32-34, 39, 45, 64, 89. Each of the IKE protocols also discloses the need to keep the nonces secure for further computations. <i>See RFC-5996</i> at 44 (Nonces), 120-123 (Security Considerations), 46-47, 52-53; <i>RFC-4306</i> at 26 (Nonces), 88 (Security Considerations), 28-29, 33. Thus, to operate on the server nonce to calculate other shared secrets and keys securely, the server random nonce must be stored in a secure memory to be protected from attackers. The alleged grounds thus render obvious to receive a server random nonce and to store the received server nonce as taught in the alleged grounds.	Initiator	Responder	-----		HDR, SAi1, KEi, Ni -->	
Initiator	Responder						

HDR, SAi1, KEi, Ni -->							

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	Exemplary disclosures include the following: <i>Bugiel</i> discloses the following at Fig. 2:  Fig. 2. SSL/TLS handshake See also <i>Bugiel</i> at 220-226, 228-236. <i>TLS-1.2</i> discloses the following at 42-43 (Server hello): When this message will be sent: The server will send this message in response to a client hello message when it was able to find an acceptable set of algorithms. If it cannot find such a match, it will respond with a handshake failure alert. Structure of this message: <pre> struct { ProtocolVersion server_version; Random random; SessionID session_id; CipherSuite cipher_suite; CompressionMethod compression_method; } ServerHello; ... </pre>

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	random This structure is generated by the server and MUST be independently generated from the ClientHello.random. <i>See also TLS-1.2</i> at 1, 4, 16-18, 20, 25-26, 33-47, 52, 55, 59, 64, 70-71, 73, 75, 77, 85, 87-88, 91-97, 100. <i>See also SSL-3.0</i> at 1, 9, 17-28, 32-35, 37-38, 41, 49-51, 55-59. <i>See also TLS-1.0</i> at 1, 4, 13-15, 21-23, 29-42, 47, 51-52, 54, 56, 64-66, 69-74. <i>See also TLS-1.1</i> at 1, 4, 14-16, 24, 31-46, 52-53, 57-58, 60, 64, 70-71, 74-80, 83. <i>See also RFC-5996</i> at 6, 9-11, 13-16, 31, 33, 37-39, 44, 46-48, 50, 52-53, 62, 69, 74, 96, 120-123, 134, 137. <i>See also RFC-4306</i> at 4, 7-8, 10, 19, 23, 26, 28-30, 32-34, 39, 45, 64, 88-89. <i>See supra</i> [1]-[2], [4]-[5], [8]-[9].
Claim 11	
[11] The method of claim 9, wherein performing the cryptographic key exchange with the remote server comprises: generating a pre-master key in the security engine; encrypting the pre-master key using the security key	The grounds alleged at the beginning of this chart disclose or suggest “performing the cryptographic key exchange with the remote server comprises: generating a pre-master key in the security engine; encrypting the pre-master key using the security key of the security engine; and sending the encrypted pre-master key to the remote server.” <i>See supra</i> [3] (“to generate a pre-master key, encrypt the pre-master key using the security key of the security engine, and send the encrypted pre-master key to the remote server.”), claim feature [1d] (“perform a cryptographic key exchange with the remote server”); <i>see also supra</i> [1], [9].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
of the security engine; and sending the encrypted pre-master key to the remote server.	
Claim 12	
[12] The method of claim 11, further comprising: wrapping the pre-master key in the security engine of the system-on-a-chip using a server public key, and storing the wrapped pre-master key in the secure memory of the security engine of the system-on-a-chip.	The grounds alleged at the beginning of this chart disclose or suggest “wrapping the pre-master key in the security engine of the system-on-a-chip using a server public key, and storing the wrapped pre-master key in the secure memory of the security engine of the system-on-a-chip.” <i>See supra</i> [1], [4] (“to wrap the pre-master key using a server public key and store the wrapped pre-master key in the secure memory”), [8]-[9], [11].
Claim 13	
[13] The method of claim 11, wherein performing the cryptographic key exchange with the remote server comprises generating the session key in the security engine of	The grounds alleged at the beginning of this chart disclose or suggest “performing the cryptographic key exchange with the remote server comprises generating the session key in the security engine of the system-on-a-chip as a function of the pre-master key.” <i>See supra</i> [1c]-[1d] (“perform a cryptographic key exchange with the remote server; generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
the system-on-a-chip as a function of the pre-master key.	session”), [5a] (“generate the session key as a function of the pre-master key”); <i>see also</i> [1], [5], [9], [11].
Claim 14	
[14] The method of claim 13, wherein generating the session key in the security engine comprises calculating, using the security engine of the system-on-a-chip, a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server.	The grounds alleged at the beginning of this chart disclose or suggest of the system-on-a-chip, a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server.” <i>See supra</i> [1c]-[1d] (“perform a cryptographic key exchange with the remote server; generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session”), [5b] (“calculate a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server”); <i>see also</i> [1], [5], [9], [11], [13].
Claim 15	
[15a] The method of claim 9 wherein performing the cryptographic key exchange with the remote server comprises: generating, in the security engine of the system-on-a-	The grounds alleged at the beginning of this chart disclose or suggest “performing the cryptographic key exchange with the remote server comprises: generating, in the security engine of the system-on-a-chip, a public Diffie-Hellman client key and a private Diffie-Hellman client key,” <i>See supra</i> [1c]-[1d] (“perform a cryptographic key exchange with the remote server; generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
chip, a public Diffie-Hellman client key and a private Diffie-Hellman client key;	session”), [6a] (“the security engine is further to generate a public Diffie-Hellman client key and a private Diffie-Hellman client key”); <i>see also supra</i> [1], [6], [9].
[15b] signing the public Diffie-Hellman client key using the security key of the security engine of the system-on-a-chip; sending the signed public Diffie-Hellman key to the remote server;	The grounds alleged at the beginning of this chart disclose or suggest “signing the public Diffie-Hellman client key using the security key of the security engine of the system-on-a-chip; sending the signed public Diffie-Hellman key to the remote server.” <i>See supra</i> [1c]-[1d] (“perform a cryptographic key exchange with the remote server; generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session”), [6b] (“sign the public Diffie-Hellman client key using the security key of the security engine, and send the signed public Diffie-Hellman key to the remote server”); <i>see also supra</i> [1], [6], [9], [15a].
[15c] receiving a public Diffie-Hellman server key from the remote server; and	The grounds alleged at the beginning of this chart disclose or suggest receiving a public Diffie-Hellman server key from the remote server. <i>See supra</i> [1c]-[1d] (“perform a cryptographic key exchange with the remote server; generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session”), [7] (“the security engine is further to receive a public Diffie-Hellman server key from the remote server”); <i>see also supra</i> [1], [9].
[15d] generating, in the security engine of the system-on-a-chip, the session key as a function of the private Diffie-	The grounds alleged at the beginning of this chart disclose or suggest “generating, in the security engine of the system-on-a-chip, the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key.”

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
Hellman client key and the public Diffie-Hellman server key.	<i>See supra</i> [1c]-[1d] (“perform a cryptographic key exchange with the remote server; generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session”), [7] (“generate the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key”); <i>see also supra</i> [1], [9], [15a]-[15c].
Claim 16	
[16a] The method of claim 9, further comprising: generating, in the security engine of the system-on-a-chip, a hash code of a preceding message sent from the client to the remote server;	The grounds alleged at the beginning of this chart disclose or suggest “generating, in the security engine of the system-on-a-chip, a hash code of a preceding message sent from the client to the remote server.” The combination alleged here implements <i>Bugiel</i>’s communications method in the security engine disclosed in [1a-1], including to implement known protocols, such as IPsec IKE or SSL/TLS Handshake Protocol messages, to establish a secure communication. The grounds alleged here implement secure protocols IKE or SSL/TLS protocols, e.g., TLS version 1.2 as <i>Bugiel</i> teaches, including the teaching in <i>Bugiel</i> and IKE or SSL/TLS protocol to perform a hash algorithm on a preceding message from the client to the server. <i>See Bugiel</i> at 225-227 (“the hash value hash(Certs)”). In addition, at the time of the '838 patent, it was well-known to a POSITA to generating a hash code. <i>See supra</i> [1a-1], [5], [14]. In addition, <i>TLS-1.2</i> teaches that the client Certificate Verify message includes a hash of prior handshake messages (the claimed “generating . . . hash code of a preceding message sent from the client to the remote server”). <i>TLS-1.2</i> at 62 (Certificate Verify). Those prior handshake messages include all handshake messages sent or received, starting at ClientHello and up to, but not including, the Certificate Verify message. <i>See TLS-1.2</i> at 62; <i>See also TLS-1.2</i> at 5-6, 16-18, 25-27, 40, 63, 66, 77, 80-81, 84, 92, 94, 97. <i>SSL-3.0</i>, <i>TLS-1.1</i> and <i>TLS-1.0</i> provide similar disclosures as <i>TLS-1.2</i>. <i>See TLS-1.1</i> at 50-51 (“CertificateVerify.signature.md5_hash MD5(handshake_messages);

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	CertificateVerify.signature.sha_hash SHA(handshake_messages)"); <i>see also TLS-1.1</i> at 12-14, 16, 24, 44, 51-53, 59, 66-67, 76, 78, 81-82; <i>TLS-1.0</i> at 45-46 ("CertificateVerify.signature.md5_hash MD5(handshake_messages); CertificateVerify.signature.sha_hash SHA(handshake_messages)"); <i>see also TLS-1.0</i> at 11-13, 15, 21-23, 41, 45-47, 52-53, 59-60, 71-72, 76-77; <i>SSL-3.0</i> at 31 ("CertificateVerify.signature.md5_hash MD5(master_secret + pad_2 + MD5(handshake_messages + master_secret + pad_1)); Certificate.signature.sha_hash SHA(master_secret + pad_2 + SHA(handshake_messages + master_secret + pad_1))"); <i>see also SSL-3.0</i> at 11, 26-27, 30-35, 39, 41, 45-46, 58, 61-62; <i>see also</i> claim feature [5b]. Therefore, in each SSL/TLS protocol, the client Certificate Verify message sent following the client key exchange message will include a hash code of the ClientHello and client key exchange messages (the claimed "a preceding message sent from the client to the remote server"), etc. Exemplary disclosures of the prior art include the following: <i>Bugiel</i> discloses the following at 225-227 (Establishing SSL/TLS connection and managing two logical SSL/TLS subchannels): Figure 2 illustrates the SSL handshake protocol. The client and server exchange ClientHello and ServerHello messages and negotiate the required system parameters. In the following steps the server's certificate Certs is sent and verified, the SSL/TLS session key Sc is generated (which is never disclosed to WalletHelper) and sealed (together with pre-master secret PMS and the hash value hash(Certs)). The resulting session-token STc is stored in a secure storage and can be loaded and unsealed later using the session identifier SID. ... All messages which are exchanged between WalletHelper and WebServer are standard messages of SSL/TLS handshake and can be found in the corresponding specification [16].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also Bugiel</i> at 235-236. <i>TLS-1.2</i> discloses the following at 62 (7.4.8. Certificate verify) When this message will be sent: This message is used to provide explicit verification of a client certificate. This message is only sent following a client certificate that has signing capability (i.e. all certificates except those containing fixed Diffie-Hellman parameters). When sent, it MUST immediately follow the client key exchange message. ... Here handshake_messages refers to all handshake messages sent or received, starting at client hello and up to, but not including, this message, including the type and length fields of the handshake messages. This is the concatenation of all the Handshake structures (as defined in Section 7.4) exchanged thus far. Note that this requires both sides to either buffer the messages or compute running hashes for all potential hash algorithms up to the time of the CertificateVerify computation. Servers can minimize this computation cost by offering a restricted set of digest algorithms in the CertificateRequest message. The hash and signature algorithms used in the signature MUST be one of those present in the supported_signature_algorithms field of the CertificateRequest message. In addition, the hash and signature algorithms MUST be compatible with the key in the client's end-entity certificate. RSA keys MAY be used with any permitted hash algorithm, subject to restrictions in the certificate, if any. <i>See also TLS-1.2</i> at 5-6, 16-18, 25-27, 40, 62-63, 66, 77, 80-81, 84, 92, 94, 97. <i>See also TLS-1.1</i> at 12-14, 16, 24, 44, 50-53, 59, 66-67, 76, 78, 81-82. <i>See also TLS-1.0</i> at 11-13, 15, 21-23, 41, 45-47, 52-53, 59-60, 71-72, 76-77.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also</i> SSL-3.0 at 11, 26-27, 30-35, 39, 41, 45-46, 58, 61-62 <i>See also</i> claim feature [5b]; <i>see also</i> [1], [5], [9], [14].
[16b] signing the hash code using the security key stored in the secure memory of the security engine; and	The grounds alleged at the beginning of this chart disclose or suggest “signing the hash code using the security key stored in the secure memory of the security engine.” The combination alleged here implements <i>Bugiel</i>’s communications method in <i>Bugiel</i>’s secure execution environment, which could include the security engine discussed in [1a-1], including to implement known protocols, such as IPsec IKE or SSL/TLS Handshake Protocol messages, etc., to establish a secure communication. The grounds alleged here implement secure protocols IKE or SSL/TLS protocols, e.g., TLS version 1.2 as <i>Bugiel</i> teaches, including the teaching in <i>Bugiel</i> and IKE or SSL/TLS protocol to sign the hash code. <i>See Bugiel</i> at 225-227. In addition, at the time of the '838 patent, it was well-known to a POSITA to performing digital signature and implement a security key. <i>See supra</i> [1a-1], [1a-3], [6], [15]. Each SSL/TLS protocol discloses signing the hash code. For example, <i>TLS-1.2</i> explains that the client Certificate Verify message includes the “digitally-signed” hash code of preceding handshake messages. <i>See TLS-1.2</i> at 62 (Certificate Verify); <i>see also TLS-1.2</i> at 33-37, 70, 74-75, 78, 92-93. <pre> Structure of this message: struct { digitally-signed struct { opaque handshake_messages[handshake_messages_length]; } } CertificateVerify; </pre> <i>TLS-1.2</i> at 62. <i>TLS-1.2</i> explains the purpose of sending the Certificate Verify message is to “provide explicit verification of a client certificate.” <i>TLS-1.2</i> at 62 (Certificate Verify).

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> also teach this claim feature. <i>See SSL-3.0</i> at 30-31 (Certificate verify), 18-21, 38, 41, 55-57; <i>TLS-1.0</i> at 45-46 (Certificate verify), 29-33, 51, 54, 69-71; <i>TLS-1.1</i> at 50-51 (Certificate verify), 31-35, 57, 60, 76. Because the purpose of signing in the TLS protocol is to allow the server to verify the identity of the client device, a POSITA thus would have found it obvious to use a key that is specific to the client to sign the hash code. Because the security key disclosed by the references as discussed in [1a-3] is encoded during manufacturing, it would have been an obvious choice to use as a device-specific key to sign the hash code, thereby allowing the server to verify the identity of the client device. A POSITA thus would have found it obvious that the client device implementing the SSL/TLS protocols using security engine discussed in [1a-1] would have signed the hash code with the security key in the secure memory as discussed in [1a-2] and [1a-3]. <i>See</i> disclosures of security key and secure memory in claim features [1a-1]-[1a-3]. As one example, <i>Chou</i> teaches such a security key stored in the secure memory of the security engine. For example, <i>Chou</i> explains that a device unique key/root key (the claimed “security key”) “may be programmed and stored” in OTP 54. <i>See Chou</i> ¶ [0038]. Further, <i>Chou</i> explains that PKA module 56 (part of the claimed “security engine”) performs digital signing by using a 1024-bit DSA (Digital Signature Algorithm) signature. <i>See Chou</i> ¶ [0045]. Because the device unique key/root key in <i>Chou</i>’s security engine is unique to the client, it would have been an obvious choice to use as a device-specific key to sign the hash code in the alleged combination because it allows the server to verify the identity of the client device. In addition to <i>Chou</i>, other references also teach the security key as discussed in [1a-3]. <i>See supra</i> [1a-3] (the references’ disclosure of security key) A POSITA thus would have found it obvious to use the grounds alleged here to sign the hash code using the security key stored in the secure memory of the security engine. <i>See</i> disclosures of security engine, security key and secure memory in claim features [1a-1]-[1a-3].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Exemplary disclosures of the prior art include the following: <i>TLS-1.2</i> discloses the following at 62 (Certificate Verify) When this message will be sent: This message is used to provide explicit verification of a client certificate. This message is only sent following a client certificate that has signing capability (i.e. all certificates except those containing fixed Diffie-Hellman parameters). When sent, it will immediately follow the client key exchange message. Structure of this message: <pre> struct { digitally-signed struct { opaque handshake_messages[handshake_messages_length]; } } CertificateVerify; </pre> Here handshake_messages refer to all handshake messages sent or received starting at client hello up to but not including this message, including the type and length fields of the handshake messages. This is the concatenation of all the Handshake structures (as defined in 7.4) exchanged thus far. Note that this requires both sides to either buffer the messages or compute running hashes for all potential hash algorithms up to the time of the CertificateVerify computation. Servers can minimize this computation cost by offering a restricted set of digest algorithms in the CertificateRequest message. The hash and signature algorithms used in the signature MUST be one of those present in the supported_signature_algorithms field of the CertificateRequest message. In addition, the hash and signature algorithms MUST be compatible with the key in the client's end-entity certificate. RSA keys MAY be used with any permitted hash algorithm, subject to restrictions in the certificate, if any.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also TLS-1.2</i> at 33-37, 62, 70, 74-75, 78, 92-93. <i>See also SSL-3.0</i> at 18-21, 30-31, 38, 41, 55-57. <i>See also TLS-1.0</i> at 29-33, 45-46, 51, 54, 69-71. <i>See also TLS-1.1</i> at 50-51, 31-35, 57, 60, 76. <i>See also Bugiel</i> at 225-227, 235-236. <i>See also [1a-1]-[1a-3]; see also [1], [6], [9], [15], [16a].</i>
[16c] sending a client certificate verification message to the remote server, the client certificate verification message including the signed hash code.	The grounds alleged at the beginning of this chart disclose or suggest “sending a client certificate verification message to the remote server, the client certificate verification message including the signed hash code.” The combination alleged here implements <i>Bugiel</i>’s communications method in <i>Bugiel</i>’s secure execution environment, which could include the security engine discussed in [1a-1], including to implement known protocols, such as IPsec IKE or SSL/TLS Handshake Protocol messages, etc., to establish a secure communication. The grounds alleged here implement secure protocols IKE or SSL/TLS protocols, e.g., TLS version 1.2 as <i>Bugiel</i> teaches, including the teaching in <i>Bugiel</i> and IKE or SSL/TLS protocol to send a client Certificate Verify message to the remote server. <i>See Bugiel</i> at 225-227. Each SSL/TLS protocol teaches sending a client Certificate Verify message, which includes the signed hash code, to the remote server. For example, <i>TLS-1.2</i> teaches generating a client Certificate Verify message, which includes hashing and signing preceding handshake messages, and sending the client Certificate Verify message to the remote server. <i>See TLS-1.2</i> at 62 (Certificate Verify); <i>see also TLS-1.2</i> at 33-37, 70, 74-75, 78, 92-93.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>SSL-3.0</i>, <i>TLS-1.0</i>, and <i>TLS-1.1</i> provide similar disclosures. <i>See SSL-3.0</i> at 30-31 (Certificate verify), 18-21, 38, 41, 55-57; <i>TLS-1.0</i> at 45-46 (Certificate verify), 29-33, 51, 54, 69-71; <i>TLS-1.1</i> at 50-51 (Certificate verify), 31-35, 57, 60, 76. The previous handshake messages include all handshake messages sent or received, starting at ClientHello and up to, but not including, the Certificate Verify message. The prior art grounds alleged here thus include sending a client Certificate Verify message, which includes the signed hash code, to the remote server. Exemplary disclosures of the prior art include the following: <i>TLS-1.2</i> discloses the following at 62 (7.4.8. Certificate verify) When this message will be sent: This message is used to provide explicit verification of a client certificate. This message is only sent following a client certificate that has signing capability (i.e. all certificates except those containing fixed Diffie-Hellman parameters). When sent, it MUST immediately follow the client key exchange message. Structure of this message: <pre> struct { digitally-signed struct { opaque handshake_messages[handshake_messages_length]; } } CertificateVerify; </pre> Here handshake_messages refer to all handshake messages sent or received starting at client hello up to but not including this message, including the type and length fields of the handshake messages. This is the concatenation of all the Handshake structures (as defined in 7.4) exchanged thus far. Note that this requires both sides to either buffer the messages or compute running hashes for all potential hash algorithms up to the time of the

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	CertificateVerify computation. Servers can minimize this computation cost by offering a restricted set of digest algorithms in the CertificateRequest message. The hash and signature algorithms used in the signature MUST be one of those present in the supported_signature_algorithms field of the CertificateRequest message. In addition, the hash and signature algorithms MUST be compatible with the key in the client's end-entity certificate. RSA keys MAY be used with any permitted hash algorithm, subject to restrictions in the certificate, if any. <i>See also TLS-1.2</i> at 33-37, 62, 70, 74-75, 78, 92-93. <i>See also SSL-3.0</i> at 18-21, 30-31, 38, 41, 55-57. <i>See also TLS-1.0</i> at 29-33, 45-46, 51, 54, 69-71. <i>See also TLS-1.1</i> at 31-35, 50-51, 57, 60, 76. <i>See also Bugiel</i> at 225-227, 235-236. <i>See supra</i> [1c]-[1d], [5], [14]; <i>see also</i> [1], [6], [9], [15], [16a]-[16b].
Claim 17	
[17pre] One or more non-transitory machine-readable storage media comprising a plurality of instructions stored thereon that, in response to execution by a computing	To the extent this preamble is limiting, the grounds alleged at the beginning of this chart disclose or suggest “[o]ne or more non-transitory machine-readable storage media comprising a plurality of instructions stored thereon that, in response to execution by a computing device, cause the computing device to” perform tasks.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
device, cause the computing device to:	The grounds alleged here uses apparatus alleged in [1pre]-[1a-3] to implement <i>Bugiel</i>'s communications method in <i>Bugiel</i>'s secure execution environment, including to implement IPsec IKE or SSL/TLS Handshake Protocol messages, to establish a secure communication. A POSITA would have understood that implementing such a combination includes using one or more non-transitory machine-readable storage media comprising a plurality of instructions stored thereon that, in response to execution by a computing device (e.g., a computing device comprising the SoC discussed in [1pre]-[1a-3]), cause the computing device to perform tasks. <i>Bugiel</i> describes, for example, “letting the trusted and untrusted code cooperate in establishing a secure channel between the user platform and the web server.” <i>Bugiel</i> at 223; <i>see also generally Bugiel</i> on mobile devices, including e.g., 219-224, 226-230, 232-235. Such applications include one or more non-transitory machine-readable storage media comprising a plurality of instructions stored thereon that, in response to execution by a computing device, cause the computing device to perform certain tasks. In addition, at the time of the '838 patent, it was well-known to a POSITA that “[o]ne or more non-transitory machine-readable storage media comprising a plurality of instructions stored thereon that, in response to execution by a computing device, cause the computing device to” perform tasks. For example, <i>Chou</i> explains that “[o]ne of ordinary skill in the art may also recognize that the functional building blocks, and other illustrative blocks, modules and components herein, may be implemented as illustrated or by discrete components, application specific integrated circuits, processors executing appropriate software and the like or any combination thereof.” <i>Chou</i> ¶ [0071]; <i>see also Chou</i> ¶ [0003] (processing device), ¶ [0005], ¶¶ [0007]-[0008], ¶¶ [0015]-[0021], ¶¶ [0023]-[0026], ¶¶ [0028]-[0032], ¶ [0038], ¶¶ [0040]-[0041], ¶¶ [0065]-[0071], Figs. 1, 3-5, claims 1-15. In addition, <i>Tkacik</i> also teaches that a process 118 (loading a key) and a process 140 (making a new key) “may be performed by controller 46 in response to the execution of software programming instructions using components of secure processing section 30, and possibly other components of host device 10.” <i>Tkacik</i> ¶ [0073], ¶ [0079]. <i>Tkacik</i> further explains that host device 10 “may be configured for any of a wide variety of different data-processing applications, including server, point-of-sale

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	terminal, wireline or wireless telephony, radio, personal computer, laptop, handheld computer, workstation, digital media player, router, modem, industrial controller, and the like.” <i>Id.</i> ¶ [0021]; <i>see also id.</i> ¶ [0008] (“Such device applications include secure high-capacity routers and servers capable of simultaneously supporting a vast number of communication sessions or other data object domains for a vast number of clients.”). Such applications include one or more non-transitory machine-readable storage media comprising a plurality of instructions stored thereon that, in response to execution by a computing device, cause the computing device to perform certain tasks. <i>See also Tkacik</i> ¶ [0007], ¶ [0024], ¶ [0030], ¶ [0034], ¶ [0037], ¶ [0043], ¶ [0053], ¶ [0058], ¶ [0064], ¶ [0066], ¶ [0081], Figs. 1-2. In addition, <i>Paaske</i> explains that the TPM includes an instruction ROM 310, an instruction multiplexer 312, and instruction cache controller 326 with an instruction cache 328. <i>See Paaske</i> ¶¶ [0051]-[0052], ¶ [0054], ¶¶ [0060]-[0061], ¶¶ [0069]-[0075], ¶ [0078], Figs. 1-3. For example, <i>Paaske</i> teaches that the instruction multiplexer 312 facilitates code retrieval from instruction ROM 310 for the processor 308 to execute. <i>Paaske</i> ¶¶ [0051]-[0052]. <i>Paaske</i> further teaches that instruction cache controller 326 may include one or more cryptographic processor(s) 334 for performing the decryption and authentication operations. <i>Paaske</i> ¶ [0054]. In addition, <i>Tugenberg</i> teaches that “[h]ost processor 16 may be one or more processing elements and is preferably the main controller for secure processing system 14. Preferably, processor 16 is a digital signal processor (DSP) or a micro-controller. On-chip system components 15 comprise other system processors, memory and other functional elements that are part of communication device 10 to allow for wireless or wireline communication of video, data and/or voice.” <i>Tugenberg</i> at 2:51-58. <i>Tugenberg</i> further explains that communication device 10 “may, for example, be a wireless communication device, such as a cellular or wireless phone, or a wireline communication device such as a computer, or a portion thereof.” <i>Id.</i> at 2:36-39; <i>see also id.</i> at 1:26-36 (“[P]ortable communication devices provide connectivity to networks such as the internet.”), 2:1-30, 4:38-49, 8:7-20. A POSITA would have understood that such applications include one or more non-transitory machine-readable storage media comprising a plurality of instructions stored thereon that, in response to execution by a

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	computing device, cause the computing device to perform certain tasks, including to perform known protocols, such as IPsec IKE or SSL/TLS, etc. to establish a secure communication session. In addition, <i>Aaraj</i> discloses “[t]he base processor considered in our experiments is a 320 MHz, five stage embedded RISC processor, with 1KB, direct-mapped instruction and data caches, and a 128-bit Processor Interface (PIF).” <i>Aaraj</i> at 8:26; <i>see also Aaraj</i> at 8:2-8:5, 8:11, 8:15-17, 8:20-8:23, 8:30-31. Such applications include one or more non-transitory machine-readable storage media comprising a plurality of instructions stored thereon that, in response to execution by a computing device, cause the computing device to perform certain tasks. In addition, <i>Gibod</i> discloses “a computer program product that comprises a non-transitory storage medium configured to store computer programs and instructions, that when executed by a processor, are configured to cause the processor to perform a method according to the instructions.” <i>Gibod</i> at ¶ [0069]; <i>see also</i> ¶ [0016], ¶¶ [0066]-[0071], claim 1. In addition, <i>Mujtaba</i> discloses “A machine-readable storage medium having instructions stored therein, which when executed by a machine, cause the machine to perform a method for operating a portable device.” <i>Mujtaba</i> at claim 19; ¶ [0065] (“a machine-readable (e.g., computer-readable) medium includes a machine (e.g., a computer) readable storage medium”). In addition, <i>Buer754</i> discloses “a variety of hardware and software processing components may be used to implement the functions.” <i>Buer754</i> at 16:36-46, 15:46-60 (“the cryptographic accelerator may have an additional command code to load and decrypt a Command Context”); <i>see also Buer754</i> at 16:8-15, Figs. 1-5, 8, 10, 14. In addition, <i>Weise</i> discloses “processing unit,” “firmware”, and “storage mechanism.” <i>Weise</i> ¶¶ [0008]-[0010], ¶¶ [0023]-[0025], ¶¶ [0028]-[0029], ¶¶ [0032]-[0035], ¶ [0037], ¶¶ [0041]-[0043], Figs. 1-5.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	Such applications as disclosed in the above references include one or more non-transitory machine-readable storage media comprising a plurality of instructions stored thereon that, in response to execution by a computing device, cause the computing device to perform certain tasks. Exemplary disclosures of the prior art include the following: <i>Bugiel</i> discloses the following at Abstract: Abstract. Mobile phones are increasingly used as general purpose computing devices with permanent Internet connection. This imposes several threats as the phone operating system (OS) is typically derived from desktop counterparts and, hence, inherits the same or similar security shortcomings. In particular, the protection of login credentials when accessing web services becomes crucial under phishing and malware attacks. On the other hand many modern mobile phones provide hardware-supported security mechanisms currently unused by most phone OSs. In this paper, we show how to use these mechanisms, in particular trusted execution environments, to protect the user's login credentials. We present the design and implementation proposal (based on Nokia N900 mobile platform) of TruWalletM, a wallet-like password manager and authentication agent towards the protection of login credentials on a mobile phone without the need to trust the whole OS software. We preserve compatibility to existing standard web authentication mechanisms. <i>See also generally Bugiel</i>, including e.g., 219-224, 226-230, 232-236. <i>See also Chou</i> ¶ [0003], ¶ [0005], ¶¶ [0007]-[0008], ¶¶ [0015]-[0021], ¶¶ [0023]-[0026], ¶¶ [0028]-[0032], ¶ [0038], ¶¶ [0040]-[0041], ¶¶ [0065]-[0071], Figs. 1, 3-5, claims 1-15. <i>See also Tkacik</i> ¶ [0073], ¶ [0079], ¶ [0021], ¶¶ [0007]-[0008], ¶ [0024], ¶ [0030], ¶ [0034], ¶ [0037], ¶ [0043], ¶ [0053], ¶ [0058], ¶ [0064], ¶ [0066], ¶ [0081], Figs. 1-2. <i>See Paaske</i> ¶¶ [0051]-[0052], ¶ [0054], ¶¶ [0060]-[0061], ¶¶ [0069]-[0075], ¶ [0078], Figs. 1-3.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also Tugenberg</i> at 2:51-58, 2:36-39, 1:26-36, 2:1-30, 4:38-49, 8:7-20. <i>See also Aaraj</i> at 8:2-8:5, 8:11, 8:15-17, 8:20-8:23, 8:26, 8:30-8:31. <i>See also Gibod</i> at ¶ [0066]-[0071], claim 1. <i>See also Mujtaba</i> at claim 19, ¶ [0065]. <i>See also Buer754</i> at 16:36-46, 15:46-60, 16:8-15, Figs. 1-5, 8, 10, 14. <i>See also Weise</i> ¶¶ [0008]-[0010], ¶¶ [0023]-[0025], ¶¶ [0028]-[0029], ¶¶ [0032]-[0035], ¶ [0037], ¶¶ [0041]-[0043], Figs. 1-5. <i>See also supra</i> [1pre], [9pre].
[17a] generate a random nonce in a security engine that is separate from a processor core of a system-on-a-chip of the computing device;	The grounds alleged at the beginning of this chart disclose or suggest to “generate a random nonce in a security engine that is separate from a processor core of a system-on-a-chip of the computing device.” <i>See supra</i> [1pre] (“A system-on-a-chip apparatus”), [1a-1] (“a system-on-a-chip comprising a security engine that is separate from a processor core of the system-on-a-chip”), [1b] (“generate a random nonce”), [9a] (“generating a random nonce in a security engine that is separate from a processor core of a system-on-a-chip of a client device”); <i>see also</i> [1], [2], [9], [10].
[17b] initiate a request for a secure communication session with a remote server over a network, the request including the random nonce;	The grounds alleged at the beginning of this chart disclose or suggest to “initiate a request for a secure communication session with a remote server over a network, the request including the random nonce.” <i>See supra</i> [1b] (“for initiating a request for a secure communication session with a remote server over a network using the nonce”), [9b] (“initiating, using the client device, a request for a secure

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	communication session with a remote server over a network, the request including the random nonce”); <i>see also</i> [1], [2], [9], [10].
[17c] perform a cryptographic key exchange, using the security engine of the system-on-a-chip, with the remote server;	The grounds alleged at the beginning of this chart disclose or suggest to “perform a cryptographic key exchange, using the security engine of the system-on-a-chip, with the remote server.” <i>See supra</i> [1c] (“perform a cryptographic key exchange with the remote server”), [9c] (“performing a cryptographic key exchange, using the security engine of the system-on-a-chip, with the remote server”); <i>see also</i> [1], [9].
[17d] generate, based on the cryptographic key exchange, a symmetric session key to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session;	The grounds alleged at the beginning of this chart disclose or suggest to “generate, based on the cryptographic key exchange, a symmetric session key to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session.” <i>See supra</i> [1d] (“generate a symmetric session key, based on the cryptographic key exchange, to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session”), [9d] (“generate a symmetric session key to encrypt messages sent to the remote server and decrypt messages received from the remote server during the secure communication session”); <i>see also</i> [1], [9].
[17e] encrypt the session key based on a security key that was encoded in a secure memory of the security engine during a manufacturing process of the system-on-a-chip;	The grounds alleged at the beginning of this chart disclose or suggest to “encrypt the session key based on a security key that was encoded in a secure memory of the security engine during a manufacturing process of the system-on-a-chip.” <i>See supra</i> [1a-2] (“has a secure memory accessible only by the security engine”), [1a-3] (“wherein the secure memory includes a security key that was encoded in the secure memory during a manufacturing process of the system-on-a-chip”), [1e] (“encrypt the symmetric session key based on the security

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	key”), [9e] (“encrypting the session key based on a security key that was encoded in a secure memory of the security engine during a manufacturing process of the system-on-a-chip”); <i>see also</i> [1], [9].
[17f] store the session key in the secure memory of the security engine of the system-on-a-chip; and	The grounds alleged at the beginning of this chart disclose or suggest to “store the session key in the secure memory of the security engine of the system-on-a-chip.” <i>See supra</i> [1a-2] (“[a security engine] has a secure memory accessible only by the security engine”), [1f] (“[the security engine to] store the encrypted session key in the secure memory”), [9f] (“storing the encrypted session key in the secure memory of the security engine of the system-on-a-chip”); <i>see also</i> [1], [9].
[17g] establish the secure communication session with the remote server using the session key.	The grounds alleged at the beginning of this chart disclose or suggest to “establish the secure communication session with the remote server using the session key.” <i>See supra</i> [1g] (“the system-on-a-chip to establish the secure communication session with the remote server over the network using the session key”), [9g] (“establishing, using the client device, the secure communication session with the remote server using the session key”); <i>see supra</i> [1], [9].
Claim 18	
[18a] The one or more non-transitory machine-readable storage media of claim 17, wherein to initiate a request for a secure communication session comprises to send a ClientHello message,	The grounds alleged at the beginning of this chart disclose or suggest “to initiate a request for a secure communication session comprises to send a ClientHello message, including the nonce, to the remote server.” <i>See supra</i> [1b] (“generate a random nonce for initiating a request for a secure communication session with a remote server over a network using the nonce”), [10a] (“initiating a request for a secure communication session comprises sending a ClientHello message, including the nonce, to the remote server”); <i>see supra</i> [1], [2], [5], [9]-[10], [14], [17].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
including the nonce, to the remote server; and	
[18b] wherein the plurality of instructions further cause the computing device to: receive a ServerHello message from the remote server, the ServerHello message including a server nonce; and store the server nonce in the secure memory of the security engine of the system-on-a-chip.	The grounds alleged at the beginning of this chart disclose or suggest to “receive a ServerHello message from the remote server, the ServerHello message including a server nonce; and store the server nonce in the secure memory of the security engine of the system-on-a-chip.” <i>See supra</i> [2] (“receive a server nonce from the remote server in a response message to the request for the secure communication session, the response message including a server nonce and store the server nonce in the secure memory”), [10b] (“receiving a ServerHello message from the remote server, the ServerHello message including a server nonce; and storing the server nonce in the secure memory of the security engine of the system-on-a-chip.”); <i>see also supra</i> [1], [4], [5], [8]-[10], [12], [14], [16]-[17], [18a].
Claim 19	
[19] The one or more non-transitory machine-readable storage media of claim 17, wherein to perform the cryptographic key exchange with the remote server comprises to: generate a pre-master key in the security engine; encrypt the pre-master key using the security key of the security engine; and	The grounds alleged at the beginning of this chart disclose or suggest “to perform the cryptographic key exchange with the remote server comprises to: generate a pre-master key in the security engine; encrypt the pre-master key using the security key of the security engine; and send the encrypted pre-master key to the remote server.” <i>See supra</i> [3] (“generate a pre-master key, encrypt the pre-master key using the security key of the security engine, and send the encrypted pre-master key to the remote server”), [1d] (“perform a cryptographic key exchange with the remote server”); <i>see also</i> [1], [4]-[5], [9], [11]-[13], [17].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
send the encrypted pre-master key to the remote server.	
Claim 20	
[20] The one or more non-transitory machine-readable storage media of claim 19, wherein the plurality of instructions further cause the computing device to: wrap the pre-master key in the security engine of the system-on-a-chip with a server public key, and store the wrapped pre-master key in the secure memory of the security engine of the system-on-a-chip.	The grounds alleged at the beginning of this chart disclose or suggest to “wrap the pre-master key in the security engine of the system-on-a-chip with a server public key, and store the wrapped pre-master key in the secure memory of the security engine of the system-on-a-chip.” <i>See supra</i> [4] (“wrap the pre-master key using a server public key and store the wrapped pre-master key in the secure memory”); <i>see also</i> [1], [3]-[5], [9], [17], [11]-[13], [19].
Claim 21	
[21] The one or more non-transitory machine-readable storage media of claim 19, wherein to perform the cryptographic key exchange with the	The grounds alleged at the beginning of this chart disclose or suggest “to perform the cryptographic key exchange with the remote server comprises to generate the session key in the security engine of the system-on-a-chip as a function of the pre-master key.”

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
remote server comprises to generate the session key in the security engine of the system-on-a-chip as a function of the pre-master key.	<i>See supra</i> [1d] (“perform a cryptographic key exchange with the remote server”), [5a] (“generate the session key as a function of the pre-master key”); <i>see also</i> [1], [3]-[5], [9], [19]-[20].
Claim 22	
[22] The one or more non-transitory machine-readable storage media of claim 21, wherein to generate the session key in the security engine comprises to calculate, using the security engine of the system-on-a-chip, a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server.	The grounds alleged at the beginning of this chart disclose or suggest “to generate the session key in the security engine comprises to calculate, using the security engine of the system-on-a-chip, a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server.” <i>See supra</i> [1d] (“generate a symmetric session key”), [5b] (“calculate a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server”); <i>see also</i> [1], [2], [5], [9]-[10], [14], [16], [17]-[18], [21].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
Claim 23	
[23a] The one or more non-transitory machine-readable storage media of claim 17, wherein to perform the cryptographic key exchange with the remote server comprises to: generate, in the security engine of the system-on-a-chip, a public Diffie-Hellman client key and a private Diffie-Hellman client key;	The grounds alleged at the beginning of this chart disclose or suggest “to perform the cryptographic key exchange with the remote server comprises to: generate, in the security engine of the system-on-a-chip, a public Diffie-Hellman client key and a private Diffie-Hellman client key.” <i>See supra</i> [1d] (“generate a symmetric session key”), [6a] (“the security engine is further to generate a public Diffie-Hellman client key and a private Diffie-Hellman client key”); <i>see also</i> [1], [6]-[8], [9], [15], [17].
[23b] sign the public Diffie-Hellman client key using the security key of the security engine of the system-on-a-chip; send the signed public Diffie-Hellman key to the remote server;	The grounds alleged at the beginning of this chart disclose or suggest to “sign the public Diffie-Hellman client key using the security key of the security engine of the system-on-a-chip; send the signed public Diffie-Hellman key to the remote server.” <i>See supra</i> [6b] (“sign the public Diffie-Hellman client key using the security key of the security engine, and send the signed public Diffie-Hellman key to the remote server”); <i>see also</i> [1], [6]-[8], [9], [15], [17].
[23c] receive a public Diffie-Hellman server key from the remote server; and	The grounds alleged at the beginning of this chart disclose or suggest to “receive a public Diffie-Hellman server key from the remote server.” <i>See supra</i> [7] (“the security engine is further to receive a public Diffie-Hellman server key from the remote server”); <i>see also</i> [1], [6]-[8], [9], [15], [17].

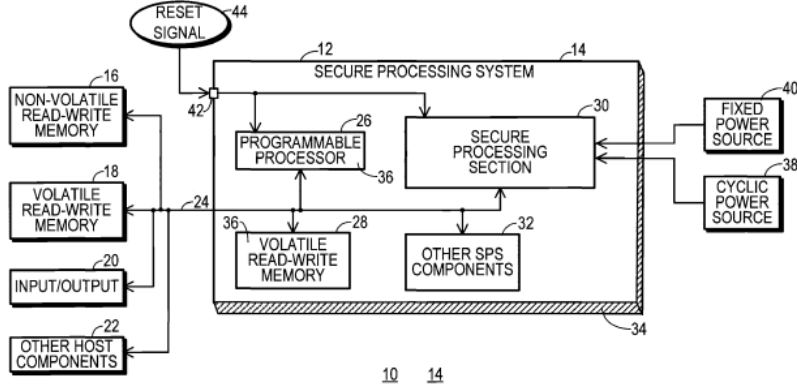
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
[23d]: generate, in the security engine of the system-on-a-chip, the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key.	The grounds alleged at the beginning of this chart disclose or suggest to “generate, in the security engine of the system-on-a-chip, the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key.” <i>See supra</i> [7] (“generate the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key”); <i>see also</i> [1], [6]-[8], [9], [15], [17].
Claim 24	
[24] The system-on-a-chip apparatus of claim 1, wherein the security engine is communicatively coupled to the processor core through an interconnect of the system-on-a-chip.	The grounds alleged at the beginning of this chart disclose or suggest “the security engine is communicatively coupled to the processor core through an interconnect of the system-on-a-chip.” For example, <i>Chou</i>’s Figures 1 and 3 show that HSM 20 and SPU-M 39/Key Algorithm Module 15 (together, the claimed “security engine”) are communicatively coupled to CPU 11/32 (the claimed “processor core”) through a Security Interface 24/42, busses 16/41 and/r 27/72 (collectively, or individually, the claimed “interconnect”). <i>See Chou</i> at Figs. 1, 3, ¶ [0020], ¶¶ [0022]-[0023], ¶ [0030], ¶ [0033], ¶¶ [0035]-[0036]; <i>see also Bugiel</i> at 230 (describing implementing the method in M-Shield of Nokia N900, where M-Shield is communicatively coupled to the processor core (e.g., ARM Cortex-A8) through an interconnect of the SoC), 219, 223-225, 234; <i>see supra</i> [1a-1].

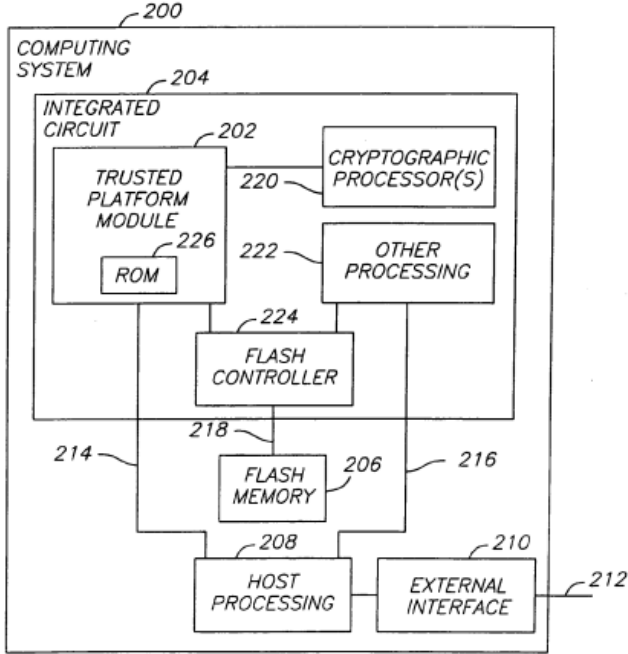
Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	FIG. 1 FIG. 3 In addition, <i>Tkacik</i>'s Figure 1 shows that secure processing section 30 (the claimed “security engine”) is communicatively coupled to the programmable processor 26 (the claimed “processor core”) through the bus 24 (the claimed “interconnect”) of the secure processing system 12 (the claimed “system-on-a-chip”). <i>See also Tkacik</i> ¶ [0020], ¶¶ [0022]-[0026], ¶ [0030], ¶¶ [0033]-[0034], ¶ [0043], ¶ [0089], Figure 2.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations
	 FIG. 1 <i>Tkacik</i> at Fig. 1. In addition, <i>Paaske</i>'s Figure 2 shows that the TPM (the "security engine") is communicatively coupled to other processing (the claimed "processor core") through busses 214 and 216 and/or flash controller 224 or host processor 208 (collectively, or individually, the claimed "interconnect"). <i>See Paaske</i> at Fig. 2, ¶¶ [0032]-[0033], ¶ [0171], ¶ [0034].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	FIG. 2  The diagram illustrates a computing system 200. It features an integrated circuit 204 which contains a trusted platform module 202. Inside the trusted platform module 202, there is a ROM 226 and a cryptographic processor 220. The trusted platform module 202 is connected to a flash controller 224, which in turn is connected to flash memory 206. The integrated circuit 204 is also connected to host processing 208 and an external interface 210. The external interface 210 is connected to an external device 212. The entire system is enclosed in a box labeled 200. In addition, <i>Tugenberg</i>'s Figure 1 shows that host processor 16 (part of the claimed "security engine") is communicatively coupled to the on-chip system components 15 (the claimed "processor core"). The on-chip system components 15 include the claimed "processor core." <i>See supra</i> [1-a1]; <i>see also Tugenberg</i> at 2:54-48 ("On-chip system components 15 comprise other system processors, memory and other functional elements that are part of communication device 10 to allow for wireless or wireline communication of video, data and/or voice."), 8:7-20. A POSITA would have understood the line joining host processor 16 and on-chip system components 15 (highlighted below in blue) is an

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	interconnects of secure processing system 14 (the claimed “system-on-a-chip”), because such a line is a typical representation of such an interconnect. <pre> graph TD subgraph 14 [SECURE PROCESSING SYSTEM] 21[21 LASER-SCRIBED ENCRYPTION KEY] --> 22[22 BLOCKING GATES] 22 --> 23[23 ENCRYPTION LOGIC CIRCUITRY] 23 --> 24[24 ZEROIZABLE MEMORY] 26[26 SECURE MEMORY] --> 25[25 SELF TEST] 25 --> 27[27 SYSTEM MONITOR] 27 --> 28[28 DEBUG DETECTOR] end 16[16 HOST PROCESSOR] <--> 15[15 ON-CHIP SYSTEM COMPONENTS] 15 <--> 12[12 EXTERNAL MEMORY] 15 <--> 13[13 COMMUNICATIONS COMPONENTS] 16 <--> 22 22 <--> 23 23 <--> 24 24 <--> 28 28 <--> 29[29 DEBUG PORT] 21 <--> 22 26 <--> 25 25 <--> 27 27 <--> 28 </pre> FIG. 1 In addition, <i>Aaraj</i> describes using SRK to encrypt migratable keys before storing them outside the TPM, demonstrating the TPM is communicatively coupled to the processor(s) outside the TPM. See <i>Aaraj</i> at 8:6. In addition, <i>Gibod</i> discloses in Fig. 5 that the SPU 518 (part of the secure processing domain) is communicatively coupled to the Processor 502 through bus 514. See <i>Gibod</i> ¶¶ [0065]-[0066]; see also Fig. 1, ¶ [0016], ¶¶ [0069]-[0070]. As discussed in claim feature [1f], <i>Gibod</i> discloses exporting encrypted keys from secure processing domain to open processing domain, demonstrating the security engine in the secure processing domain is communicatively coupled to the processor core in the public processing domain. See <i>supra</i> [1f].

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	In addition, Fig. 1 of <i>Mujtaba</i> illustrates the wireless communication processor 101 is communicatively coupled to application processor 102. <i>See also Mujtaba</i> at Fig. 4, ¶¶ [0026]-[0033], ¶¶ [0042]-[0044]. In addition, Figure 3 of <i>Buer754</i> at Fig. 3 illustrates security module 320 and cryptographic accelerator 326 (together forming the security engine) are communicatively coupled to host processor through interconnect); <i>see also Buer754</i> at 6:13-23, 16:35-62 (describing the components may be combined in one integrated circuits and the connections in the drawing may be in an integrated circuit), 4:61-7:34, 10:14-11:3, 17:48-62, Figs. 1-3, 10. In addition, <i>Weise</i> also teaches its HSM is communicatively coupled to the processor core through an interconnect of the system-on-a-chip by disclosing “exporting keys outside the HSM” and “HSM available from various vendors can be connected to computer systems individually or in a cluster to provide cryptographic processing used by the computer systems.” <i>Weise</i> ¶ [0009], ¶¶ [0029] (“The host bus adapter 230 enables the composite HSM cryptographic accelerator to securely communicate with other portions of the host or network as appropriate.”); <i>see also id.</i> Abstract, ¶¶ [0012]-[0014], ¶¶ [0028]-[0029], ¶¶ [0032]-[0033], ¶ [0035], ¶¶ [0041]-[0043], Figs. 1-3, 5. The above disclosures are consistent with the specification of the '838 patent, which describes an “interconnect such as a bus, point-to-point, or other interconnect capable of facilitating communication between the various components of the SOC.” '838 pat. at 3:67-4:3. Exemplary disclosures of the prior art include the following: <i>Chou</i> discloses the following at ¶ [0020]: Aside from HSM 20, device 10 may have a variety of circuits, components and/or devices. FIG. 1 shows one example system 10 in which a number of components are shown. The shown device 10 includes a processor 11 (shown as a central processing unit or CPU), DMA (direct memory access) component 12, ROM (read-only-memory) 13, RAM (random-access-memory) 14, which are all coupled to bus 16. Other components, such as

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	a memory controller, cache memory, bus controller and interfaces, bridges, etc. are not shown, but may be present in other embodiments. The operation and function of components 11-14 are generally known and these components are generally present in many computing devices, including computers, wired and wireless devices, mobile phones, set-top boxes, routers, servers, as well as other devices. As noted in FIG. 1, components 11-14 reside outside of a boundary established by HSM 20 and, therefore, are regarded as residing in a non-secure domain or zone. In one embodiment, HSM 20 is constructed on a single integrated circuit chip. In another embodiment, device 10, including HSM 20, is constructed on a single integrated circuit chip. <i>Chou</i> discloses the following at ¶¶ [0022]-[0023]: For HSM 20, all data transfers between the secure domain and non-secure (unsecure) domain is routed through security interface 24. As shown, a connection or bus 27 couples security interface 24 to bus 16, so that all accesses to HSM 20 and data transfers into and out of HSM 20 are routed through security interface 24. No other accesses are permitted from the non-secure domain to HSM 20, except for the key cache operation described later in the description. In operation, when encrypted data is received by device 10, components within the non-secure domain, such as CPU 11, process the data and transfers the encrypted data to security interface 24. Security interface 24 is utilized as the single interface for data transfers between components of HSM 20 and devices that reside outside of HSM 20. <i>See also Chou</i> at Figs. 1, 3, ¶ [0020], ¶¶ [0022]-[0023], ¶ [0030], ¶ [0033], ¶¶ [0035]-[0036]. <i>See also Bugiel</i> at 219, 223-225, 230, 234-236. <i>See also Tkacik</i> ¶ [0020], ¶¶ [0022]-[0026], ¶ [0030], ¶¶ [0033]-[0034], ¶ [0043], ¶ [0089], Figs. 1 and 2.

Appendix 838-02: Invalidity of U.S. Patent No. 9,887,838 (the '838 Patent) Based on *Bugiel*

The '838 Patent	<u>Exemplary Disclosures Relevant to <i>Bugiel</i> and Related Combinations</u>
	<i>See also Paaske</i> at Fig. 2 and 3, ¶¶ [0032]-[0033], ¶ [0171], ¶ [0034]. <i>See also Tugenberg</i> at 2:54-48, 8:7-20, Fig. 1. <i>See also Aaraj</i> at 8:6, 8:30-8:31. <i>See also Gibod</i> ¶ [0016], ¶¶ [0065]-[0066], ¶¶ [0069]-[0070], Fig. 1 and 5. <i>See also Mujtaba</i> at Figs. 1 and 4, ¶¶ [0026]-[0033], ¶¶ [0042]-[0044]. <i>See also Buer754</i> at 6:13-23, 16:35-62, 4:61-7:34, 10:14-11:3, 17:48-62, Figs. 2-3, 10. <i>See also Weise</i> Abstract, ¶ [0009], ¶¶ [0012]-[0014], ¶¶ [0028]-[0029], ¶¶ [0032]-[0033], ¶ [0035], ¶¶ [0041]-[0043], Figs. 1-3, 5. <i>See also</i> disclosures in [1pre] (“A system-on-a-chip apparatus”), and [1a-1] (“a system-on-a-chip comprising a security engine that is separate from a processor core of the system-on-a-chip”); <i>see also</i> claim 1.