

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SAMSUNG ELECTRONICS CO., LTD.; AND
SAMSUNG ELECTRONICS AMERICA, INC.,

Petitioner

v.

DAEDALUS PRIME LLC
Patent Owner.

Case No. IPR2023-00727
U.S. Patent No. 9,887,838

**PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 9,887,838**

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. MANDATORY NOTICES UNDER 37 C.F.R. §42.8.....	1
III. FEE AUTHORIZATION	3
IV. GROUNDS FOR STANDING.....	3
V. RELIEF REQUESTED	3
VI. THE CHALLENGED PATENT	4
VII. LEVEL OF ORDINARY SKILL IN THE ART	7
VIII. PRIORITY DATE	8
IX. CLAIM CONSTRUCTION	8
X. BRIEF DESCRIPTION OF THE APPLIED PRIOR ART REFERENCES	8
A. Chou (Ex-1008).....	8
B. Buer (Ex-1009).....	11
C. TLS-1.0 (Ex-1006)	12
XI. DETAILED EXPLANATION OF THE UNPATENTABILITY GROUNDS	13
A. Ground 1: Chou in view of knowledge of a POSITA (Claims 1, 9, 17, 24).....	13
1. Independent Claim 1	13
a. 1[pre]: A system-on-a-chip apparatus comprising:	13
b. 1[a] a system-on-a-chip comprising a security engine that is separate from a processor core of the system-on-a-chip and has a secure memory accessible only by the security engine, 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 security engine to:.....	14
c. 1[b] generate a random nonce for initiating a request for a secure communication session with a remote server over a network using the nonce;	18

TABLE OF CONTENTS

(continued)

	Page
d. 1[c] perform a cryptographic key exchange with the remote server;	20
e. 1[d] 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;	21
f. 1[e] encrypt the symmetric session key based on the security key; and	23
g. 1[f] store the encrypted session key in the secure memory, ...	23
h. 1[g] the system-on-a-chip to establish the secure communication session with the remote server over the network using the session key.	25
2. Claim 24	26
a. 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.	26
3. Independent Claims 9, 17	27
a. 9[pre] A method comprising;. 17[pre] 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:	27
b. 9[a] 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; 17[a] 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;	28

TABLE OF CONTENTS

(continued)

	Page
c. 9[b] 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; 17[b] initiate a request for a secure communication session with a remote server over a network, the request including the random nonce;.....	28
d. 9[c] performing a cryptographic key exchange , using the security engine of the system-on-a-chip, with the remote server; 17[c] perform a cryptographic key exchange, using the security engine of the system-on-a-chip, with the remote server;	28
e. 9[d] 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; 17[d] 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;.....	29
f. 9[e] 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; 17[e] 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;.....	29
g. 9[f] storing the encrypted session key in the secure memory of the security engine of the system-on-a-chip; and 17[f] store the session key in the secure memory of the security engine of the system-on-a-chip; and.....	29

TABLE OF CONTENTS

(continued)

	Page
h. 9[g] establishing, using the client device, the secure communication session with the remote server using the session key. 17[g] establish the secure communication session with the remote server using the session key.....	30
B. Ground 2: Chou, in combination with TLS-1.0	30
1. Motivation to Combine	30
2. Dependent Claim 3	32
a. 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.....	32
3. Dependent Claim 4	33
a. 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.	33
4. Dependent Claim 5	35
a. 5[pre]/5[a]: 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	35
b. 5[b] calculate a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server	36
5. Dependent Claim 6	37
a. 6[pre]/6[a]. 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, ...	37
b. 6[b] sign the public Diffie-Hellman client key using the security key of the security engine, ...	38

TABLE OF CONTENTS

(continued)

	Page
c. 6[c] send the signed public Diffie-Hellman key to the remote server.....	39
6. Dependent Claim 7	39
a. 7[a] 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.....	39
b. 7[b] generate the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key.	40
7. Dependent Claims 10, 18.....	41
a. 10[pre]. 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 further comprising: 18[pre]. 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, including the nonce, to the remote server; and.....	41
b. 10[a] receiving a ServerHello message from the remote server, the ServerHello message including a server nonce; and 18[a] 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.....	42
c. 10[b] storing the server nonce in the secure memory of the security engine of the system-on-a- chip. 18[b] store the server nonce in the secure memory of the security engine of the system-on-a- chip.....	43
8. Dependent Claims 11, 19.....	43

TABLE OF CONTENTS

(continued)

	Page
a. 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 of the security engine; and sending the encrypted pre-master key to the remote server. 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 send the encrypted pre-master key to the remote server.....	43
9. Dependent Claims 12, 20	44
a. 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. 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.....	44
10. Dependent Claims 13, 21	44

TABLE OF CONTENTS

(continued)

	Page
a. 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 system-on-a-chip as a function of the pre-master key. 21. The one or more non-transitory machine-readable storage media of claim 19, wherein 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.....	44
11. Dependent Claims 14, 22	45
a. 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. 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.....	45
12. Claims 15, 23	45
a. 15[pre]/15[a] 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-chip, a public Diffie-Hellman client key and a private Diffie-Hellman client key; 23[pre]/23[a] 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: sign the public Diffie-Hellman client key using the security key of the security engine.....	45

TABLE OF CONTENTS

(continued)

	Page
b. 15[b] signing the public Diffie-Hellman client key using the security key of the security engine of the system-on-a-chip; 23[b] sign the public Diffie-Hellman client key using the security key of the security engine of the system-on-a-chip;.....	46
c. 15[c] sending the signed public Diffie-Hellman key to the remote server; 23[c] send the signed public Diffie-Hellman key to the remote server;.....	46
d. 15[d] receiving a public Diffie-Hellman server key from the remote server; and 23[d] receive a public Diffie-Hellman server key from the remote server; and.....	46
e. 15[e] 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. 23[e] 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.....	46
C. Grounds 3 and 4: Buer in view of knowledge of a POSITA or in view of Chou (Claims 1, 3-7, 9-15, 17-24).....	47
1. Motivation to Combine Buer with Chou	48
2. Independent Claim 1	50
a. 1[pre].....	50
b. 1[a].....	50
c. 1[b].....	52
d. 1[c].....	54
e. 1[d].....	54
f. 1[e].....	56
g. 1[f]	56
h. 1[g].....	60
3. Dependent Claim 3	61

TABLE OF CONTENTS

(continued)

	Page
4. Dependent Claim 4	62
5. Dependent Claim 5	65
a. 5[pre]/5[a]	65
b. 5[b]	65
6. Dependent Claim 6	66
a. 6[pre]/6[a]	66
b. 6[b]	67
c. 6[c]	68
7. Dependent Claim 7	68
a. 7[pre]/7[a]	68
b. 7[b]	69
8. Claim 24	70
9. Independent Claims 9, 17	71
a. 9[pre], 17[pre]	71
b. 9[a], 17[a]	71
c. 9[b], 17[b]	72
d. 9[c], 17[c]	72
e. 9[d], 17[d]	72
f. 9[e], 17[e]	72
g. 9[f], 17[f]	72
h. 9[g], 17[g]	72
10. Dependent Claims 10, 18	72
a. 10[pre], 18[pre]	72
b. 10[a], 18[a]	73
c. 10[b], 18[b]	74
11. Dependent Claims 11, 19	76
12. Dependent Claims 12, 20	76
13. Dependent Claims 13, 21	76

TABLE OF CONTENTS
(continued)

	Page
14. Dependent Claims 14, 22	76
15. Claims 15, 23	76
a. 15[pre]/15[a], 23[pre]/23[a].....	76
b. 15[b], 23[b]	76
c. 15[c], 23[c]	76
d. 15[d], 23[d]	76
e. 15[e], 23[e]	76
D. Grounds 5 and 6: Grounds 3 and 4, in further combination with TLS-1.0.....	77
1. Motivation to Combine	77
2. Dependent Claim 3	78
3. Dependent Claim 4	79
4. Dependent Claim 5	79
5. Dependent Claim 7	79
6. Dependent Claims 10, 18	80
a. 10[pre], 18[pre].....	80
b. 10[a], 18[a]	80
c. 10[b], 18[b]	80
7. Dependent Claims 11, 19.....	80
8. Dependent Claims 12, 20	80
9. Dependent Claims 13, 21	81
10. Dependent Claims 14, 22	81
11. Claims 15, 23	81
XII. THE BOARD SHOULD NOT USE ITS DISCRETION TO DENY INSTITUTION UNDER <i>FINTIV</i>	81
XIII. CONCLUSION.....	85

LIST OF EXHIBITS¹

Ex-1001	U.S. Patent No. 9,887,838 (“the ’838 Patent”)
Ex-1002	Declaration of Dr. Mark E. Crovella
Ex-1003	Curriculum Vitae of Dr. Mark E. Crovella
Ex-1004	Prosecution History of the 838 Patent
Ex-1005	W. Diffie and M. E. Hellman, <i>New Directions in Cryptography</i> , IEEE Transactions on Information Theory, IEEE Transactions on Information Theory, Vol. 22, No. 6, 644-654 (1976)
Ex-1006	T. Dierks and 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”)
Ex-1007	R.L. Rivest, A. Shamir, and L. Adleman, <i>A Method for Obtaining Digital Signatures and Public-Key Cryptosystems</i> , Communications of the ACM, Vol. 21, 120–126 (1978) (the “RSA Paper”)
Ex-1008	United States Patent Application Publication No. 2011/0191599 (“Chou”)
Ex-1009	U.S. Patent No. 7,773,754 (“Buer”)
Ex-1010	Declaration of Sandy Ginoza for IETF, RFC 2246: The TLS Protocol Version 1.0, RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 (the “IETF Declaration”)
Ex-1011	U.S. Patent No. 4,405,829 (“Rivest”)

¹ Four-digit pin citations beginning with 0 are to the branded numbers added by Samsung in the bottom right corner of the exhibits. All other pin citations are to original page, column, paragraph, or line numbers.

I. INTRODUCTION

Samsung Electronics Co., Ltd. and Samsung Electronics America, Inc., (collectively, “Petitioner”) request *inter partes* review (“IPR”) of Claims 1, 3-7, 9-15, 17-24 of U.S. Patent No. 9,887,838 (“the ’838 Patent”) (Ex-1001), currently assigned to Daedalus Prime LLC (“PO”).

II. MANDATORY NOTICES UNDER 37 C.F.R. §42.8

Real Parties-in-Interest: Petitioner identifies the following real parties-in-interest: Samsung Electronics Co., Ltd., Samsung Electronics America, Inc., Samsung Semiconductor, Inc., Samsung Austin Semiconductor, LLC, and Arm Ltd.

Related Matters: PO has asserted the ’838 Patent against Samsung Electronics Co., Ltd., Samsung Electronics America, Inc., Samsung Semiconductor, Inc., and Samsung Austin Semiconductor, LLC in Daedalus Prime LLC v. Samsung Electronics Co., Ltd., No. 2:22-cv-00354 (E.D. Tex. Sept. 12, 2022).

Lead and Back-Up Counsel:

- Lead Counsel:

John Kappos (Reg. No. 37,861)
O’Melveny & Myers LLP
2501 North Harwood Street, Suite 1700
Dallas, TX 75201
Telephone: (972) 360-1900
Fax: (972) 360-1901

E-Mail: jkappos@omm.com

- First Backup Counsel:

Benjamin M. Haber (Reg. No. 67,129)
O'Melveny & Myers LLP
400 South Hope Street, 18th Floor
Los Angeles, CA 90071
Telephone: (213) 430-6000
Fax: (213) 430-6407
E-Mail: bhaber@omm.com

- Additional Backup Counsel:

Xin-Yi Zhou (Reg. No. 63,366)
O'Melveny & Myers LLP
400 S. Hope Street
Los Angeles, CA 90071
Telephone: (213) 430-6000
Fax: (213) 430-6407
E-Mail: vzhou@omm.com

Jeff Baxter (Reg. No. 45,560)
O'Melveny & Myers LLP
2501 North Harwood Street, Suite 1700
Dallas, TX 75201
Telephone: (972) 360-1900
Fax: (972) 360-1901
E-Mail: jbaxter@omm.com

William M. Fink (Reg. No. 72,332)
O'Melveny & Myers LLP
1625 Eye Street, NW
Washington, DC 20006
Telephone: (202) 383-5300
Fax: (202) 383-5414
E-Mail: tfink@omm.com

Service Information: Petitioner consents to electronic service by email to

the following addresses:

- jkappos@omm.com
- bhaber@omm.com
- vzhou@omm.com
- jbaxter@omm.com
- tfink@omm.com
- OMMSAMSUNGDAEDALUSEDTX@omm.com

III. FEE AUTHORIZATION

Pursuant to 37 C.F.R. §42.15(a) and §42.103(a), the PTO is authorized to charge any and all fees to Deposit Account No. LA500639.

IV. GROUNDS FOR STANDING

Petitioner certifies that the '838 Patent is available for review, this Petition is timely filed, and Petitioner is not barred or estopped from requesting review.

V. RELIEF REQUESTED

Petitioner requests review and cancellation of Claims 1, 3-7, 9-15, 17-24 as unpatentable based on the following grounds, supported by a declaration from Dr. Mark E. Crovella. Ex-1002 ¶28; Ex-1003.

Ground	Summary
1	Claims 1, 9, 17, 24 are obvious over Chou (Ex-1008)

Ground	Summary
2	Claims 3-7, 10-15, 18-23 are obvious over Chou (Ex-1008) in view of TLS-1.0 (Ex-1006)
3	Claims 1, 3-7, 9-15, 17-24 are obvious over Buer (Ex-1009)
4	Claims 1, 3-7, 9-15, 17-24 are obvious over Buer (Ex-1009) in view of Chou (Ex-1008)
5	Claims 3-5, 7, 10-15, 18-23 are obvious over Buer (Ex-1009) in view of TLS-1.0 (Ex-1006)
6	Claims 3-5, 7, 10-15, 18-23 are obvious over Buer (Ex-1009) in view of Chou (Ex-1008) and TLS-1.0 (Ex-1006)

VI. THE CHALLENGED PATENT

The '838 Patent describes a “hardware security engine” used to perform a cryptographic key exchange according to the known SSL protocol. Ex-1001, Abstract. While much of the specification describes specific communication steps by the security engine, those steps are not the alleged invention of the '838 Patent. Rather, those steps, such as shown in Figure 3 (reproduced below) and

accompanying descriptions, are the standard description of the SSL/TLS protocol.

Ex-1002 ¶¶61-70.

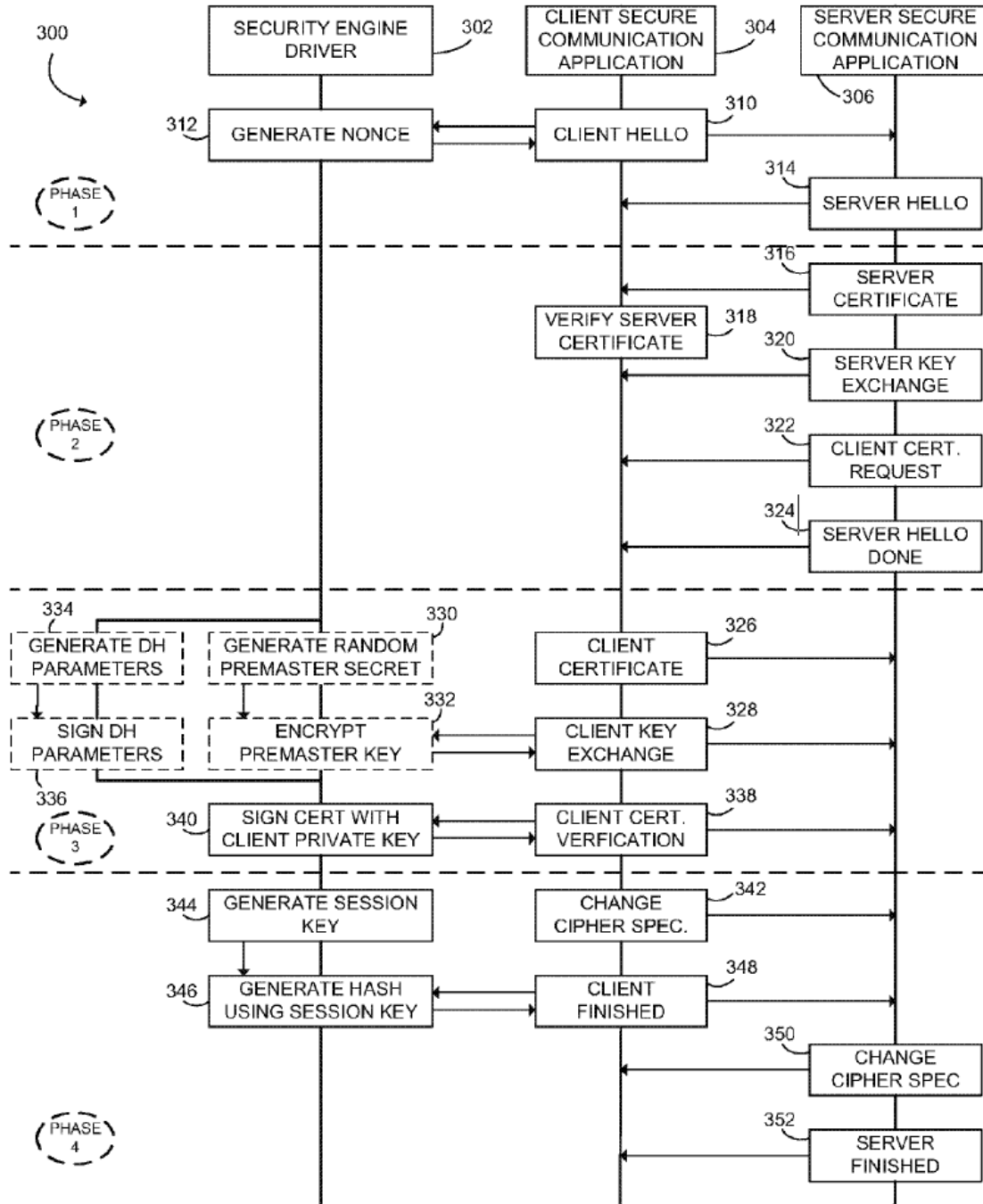


FIG. 3

Ex-1001, Fig. 3.

Each of the steps in the specification matches the standard SSL/TLS protocol. Ex-1002 ¶¶79-80. These steps (i.e., Client Hello message, which includes a random nonce initiating the key exchange; generating a pre-master key; creating a symmetric session key from the pre-master key) are performed by a “security engine” (highlighted yellow). The “security engine” includes a “secure memory” (green) *Id.* ¶¶107. The security engine also includes a “security key” (cyan) which is a device-specific key encoded at the time of manufacture of the SoC that includes the security engine. *Id.* The “secure memory” stores data processed by the security engine, such as the session key encrypted by the “security key,” the encrypted pre-master key, or other data received during the SSL key exchange procedure (e.g., a server nonce received from the server).

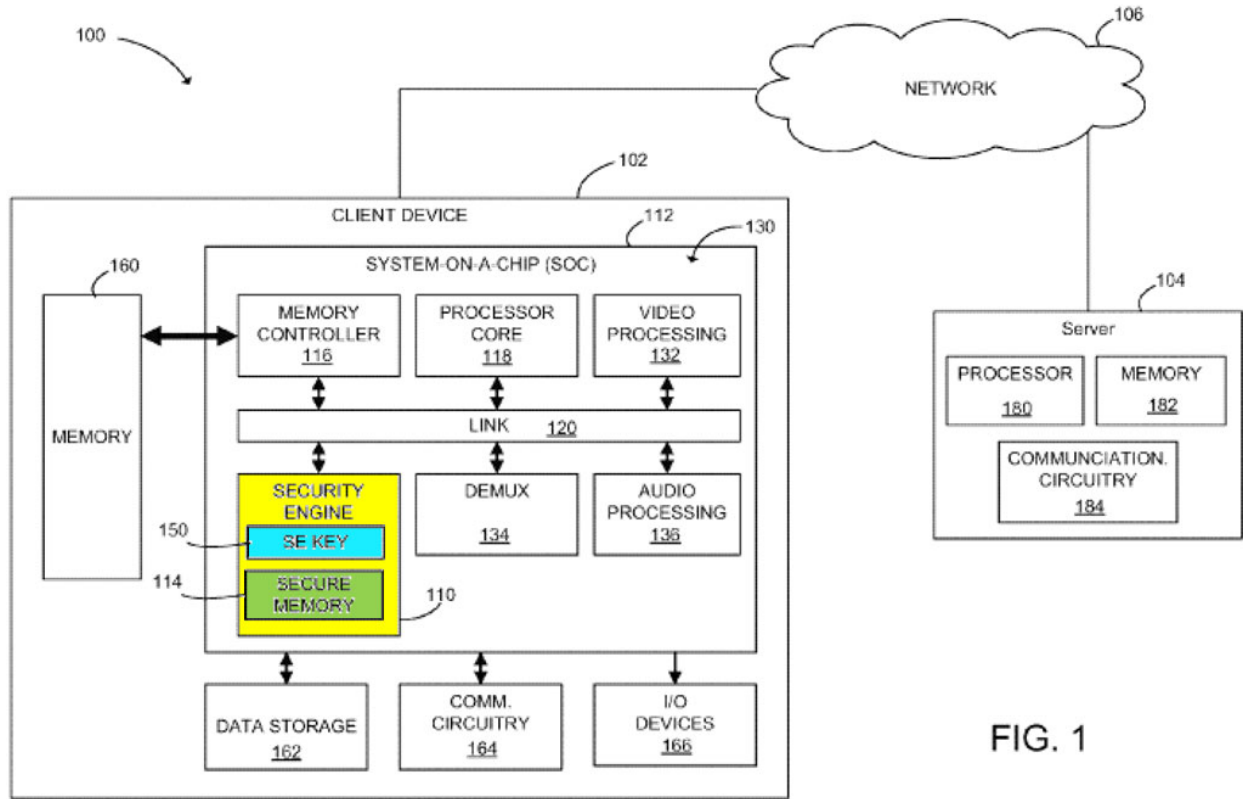


FIG. 1

Ex-1001, Fig. 1 (highlight added).²

VII. LEVEL OF ORDINARY SKILL IN THE ART

A person of ordinary skill in the art at the relevant time (“POSITA”) would have had a bachelor’s degree in electrical engineering, computer engineering, computer science, or a related field, and at least two years of experience in the research, design, development, and/or testing of secure communication and/or security methods, and related firmware and software, or the equivalent, with additional education substituting for experience and vice versa. Ex-1002 ¶¶25-27.

² Unless otherwise noted, all colored highlights and annotations are added by the petitioner.

VIII. PRIORITY DATE

For purposes of this proceeding only, Petitioner assumes the '838 Patent is entitled to its earliest alleged priority date, December 15, 2011, and thus applies pre-AIA 35 U.S.C. § 102.

IX. CLAIM CONSTRUCTION

Petitioner interprets the claims of the '838 Patent according to 37 C.F.R. § 42.100(b). To resolve the particular grounds presented in this Petition, Petitioner does not believe any term requires explicit construction. Ex-1002 ¶18.³

X. BRIEF DESCRIPTION OF THE APPLIED PRIOR ART REFERENCES

Each of the below prior art relied on in this Petition was not considered by the PTO during prosecution of the '838 Patent. Ex-1001, Cover.

A. Chou (Ex-1008)

Chou was filed 2/26/10, published 8/4/11, and qualifies as prior art under at least pre-AIA 35 U.S.C. §§102(a), (e).

Chou is directed to a “hardware security module that provides a secure boundary for retention of a secure key within the secure boundary and prevention of unauthorized accesses from external sources outside of the secure boundary to

³ Claim construction procedures have not yet begun in the related district court matter. Petitioner reserves the right to revisit and revise these constructions in district court as necessary. Additionally, Petitioner will request leave to submit any claim construction order as soon as it becomes available, so that it is timely made of record in the proceeding and can be considered by the Board. 37 C.F.R. § 42.100(b).

obtain the secure key.” Ex-1008, Abstract. To ensure only authorized parties can access transmitted data, the data is encrypted using a secure key. *See* Ex-1008, ¶6; Ex-1002 ¶¶88-89. But because the security of the data can be guaranteed only if the key remain secure, Chou “provide[s] a more secure key management scheme where a secure key may be unwrapped by a recipient, but the key not be revealed to the recipient and/or unauthorized resources.” Ex-1008 ¶9. This is accomplished by a “hardware security module” (HSM). Ex-1002 ¶89.

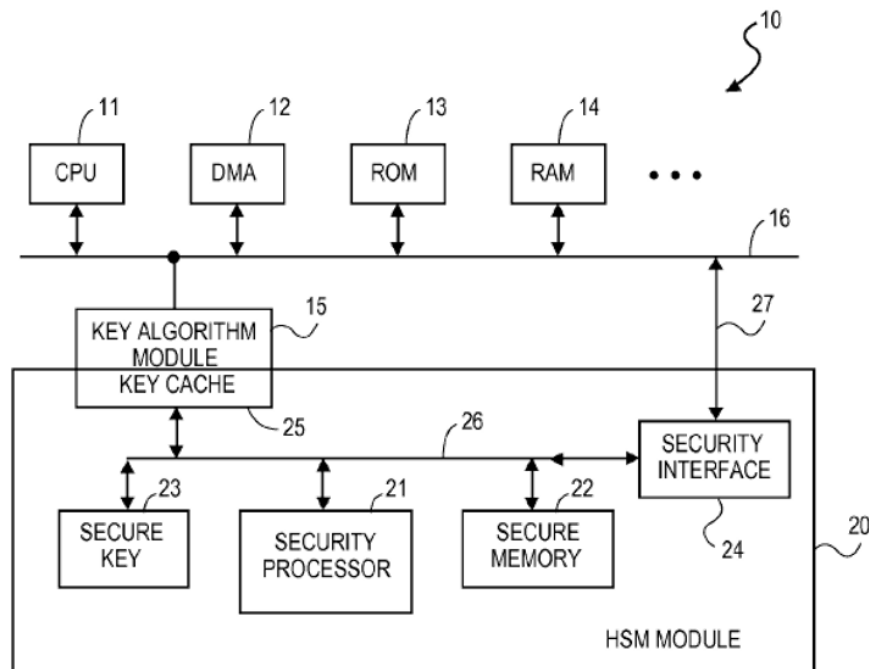


FIG. 1

Ex-1008, Fig. 1.

Chou’s HSM includes its own security processor separate from the CPU of the system. Ex-1008 ¶20; Fig. 1; Ex-1002 ¶90. The HSM also includes a secure memory inside the HSM , which can only be accessed by the HSM itself. Ex-1002

¶91. The secure memory stores the data processed (or to be processed) by the security processor inside the HSM, thereby protecting the sensitive data from unauthorized access. In addition, to protect the secure key, the HSM encrypts the secure key (such as the session key used to encrypt and decrypt the data exchanged during the exemplary secure financial transaction) using a device unique key inside a “secure boundary” (highlight in red below) of the HSM. *Id.* ¶92.

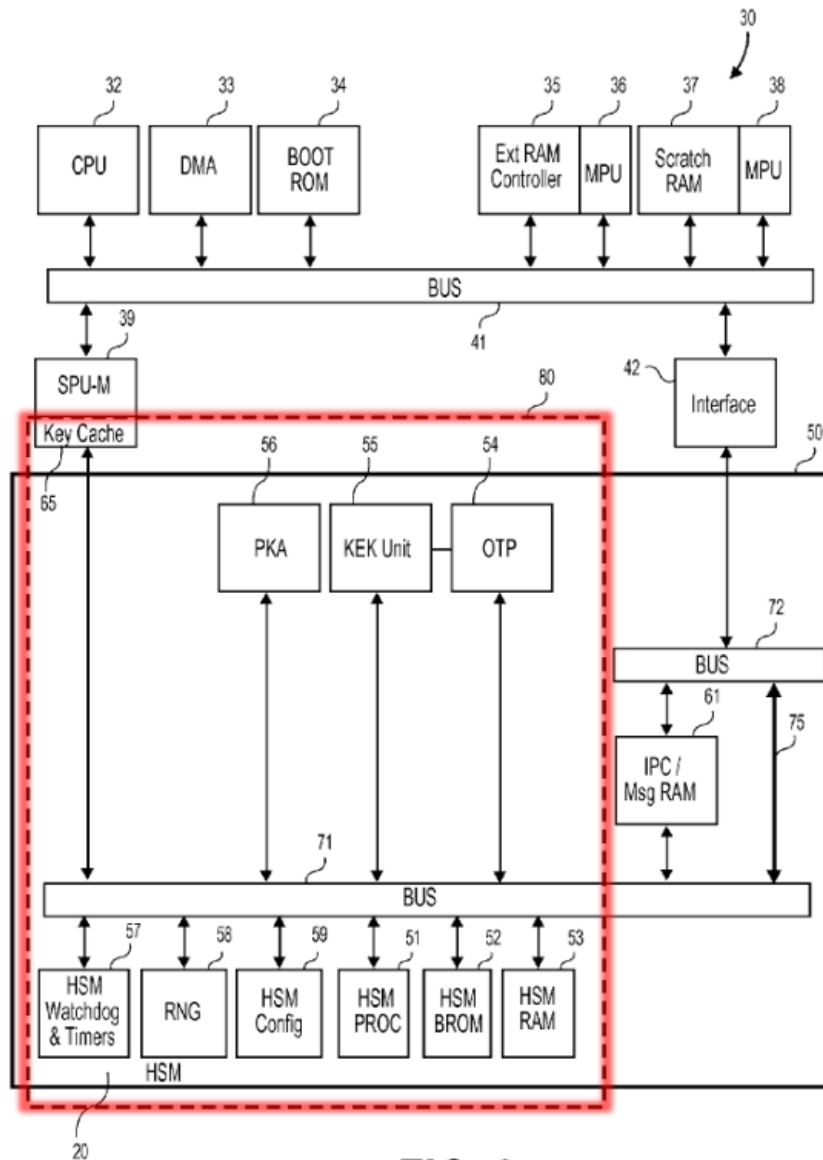


FIG. 3

Ex-1008, Fig. 3.

B. Buer (Ex-1009)

Buer was filed 7/8/02, published as US 2004/0005061A1 1/8/04, and qualifies as prior art under at least pre-AIA 35 U.S.C. §§102(a), (b).

Buer described the same subject matter as the '838 Patent—i.e., “[m]ethods and systems...for providing secured data transmission and for managing cryptographic keys.” Ex-1009, Abstract. Buer discloses hardware structure for secure key management as well as the communication steps for establishing symmetric keys between two devices over a network. Ex-1002 ¶¶94.

Buer also disclose a security module with its own processor, separate from the host processor of the system. Ex-1002 ¶¶95. Buer’s Figure 13, reproduced below, shows an embodiment showing the communication between a main security module (in a host device) and a remotely located satellite security module (a remote server). *Id.*

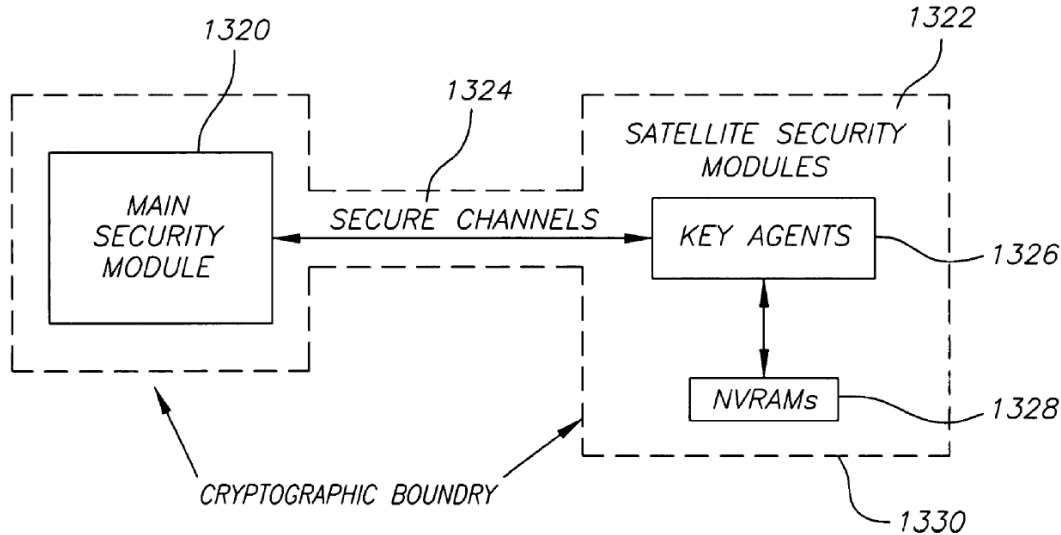


FIG. 13

Ex-1009, Fig. 13.

Buer also discloses cryptographic key exchange algorithms for establishing a session key to encrypt the data exchanged between the main and satellite security modules. Ex-1002 ¶96. Once the main security module generates the symmetric session key, the session key is encrypted based on the Key Encryption Key (KEK), encoded during manufacturing of the security module. Ex-1002, ¶96; Ex-1009, 5:43-51; 6:41-43; *see also* 1:9-12.

C. TLS-1.0 (Ex-1006)

Dierks and Allen, The TLS Protocol Version 1.0 (“TLS-1.0”) was published in January 1999 by Internet Engineering Task Force (IETF), and qualifies as prior art under at least pre-AIA 35 U.S.C. §§102(a), (b). Ex-1010 ¶¶6-11.

Transport Layer Security (TLS) is a protocol that provides secure communication over the Internet. The TLS protocol provides the framework and mechanisms for secure communication. Ex-1002 ¶¶61-68. TLS-1.0 describes the TLS Handshake Protocol, a procedure by which a client and server can establish a secure communication session. *Id.* ¶99. The session keys are generated during the TLS handshake and are used to encrypt and decrypt data transmitted between the parties. *Id.* ¶¶67-68. This process ensures that the communication is protected from eavesdropping, tampering, and impersonation, and that the authenticity of the client and server can be verified. *Id.* ¶68.

XI. DETAILED EXPLANATION OF THE UNPATENTABILITY GROUNDS

A. Ground 1: Chou in view of knowledge of a POSITA (Claims 1, 9, 17, 24)

1. Independent Claim 1

a. 1[pre]: A system-on-a-chip apparatus comprising:

Chou discloses a hardware security module (HSM) implemented on a system-on-a-chip apparatus, as shown in Figure 4. Ex-1002 ¶104; Ex-1008 ¶65 (components of device implemented in “a complete system on the IC chip (System-On-Chip or SOC)”).

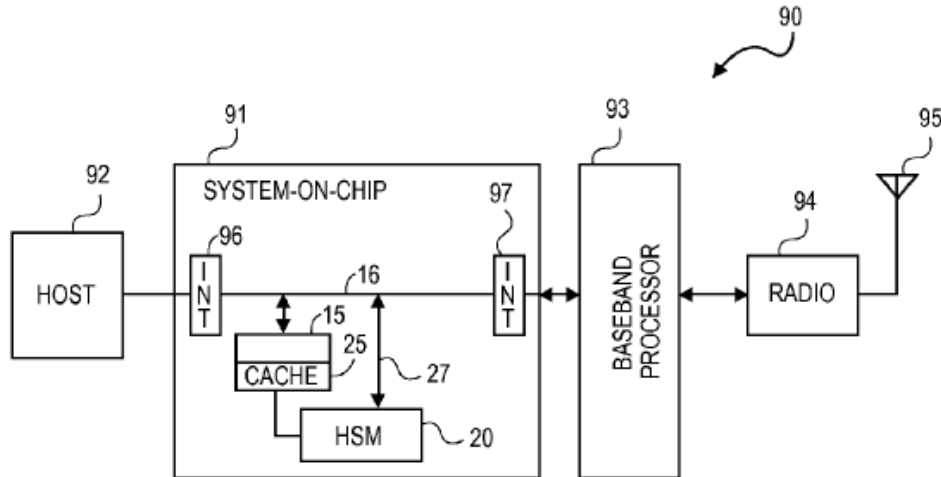


FIG. 4

- b. **1[a] a system-on-a-chip comprising a security engine that is separate from a processor core of the system-on-a-chip and has a secure memory accessible only by the security engine, 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 security engine to:**

Chou's system-on-a-chip comprises a hardware security module ("HSM") and a Secure Processing Unit for Mobile processors (SPU-M, also called "key algorithm module"), together forming a security engine. *See* Ex-1008 ¶54; Fig. 3. Figure 3 shows the HSM and the SPU-M (the security engine, highlighted yellow) are separate from the processor core of the system (the CPU **32**, blue). Ex-1002 ¶105.

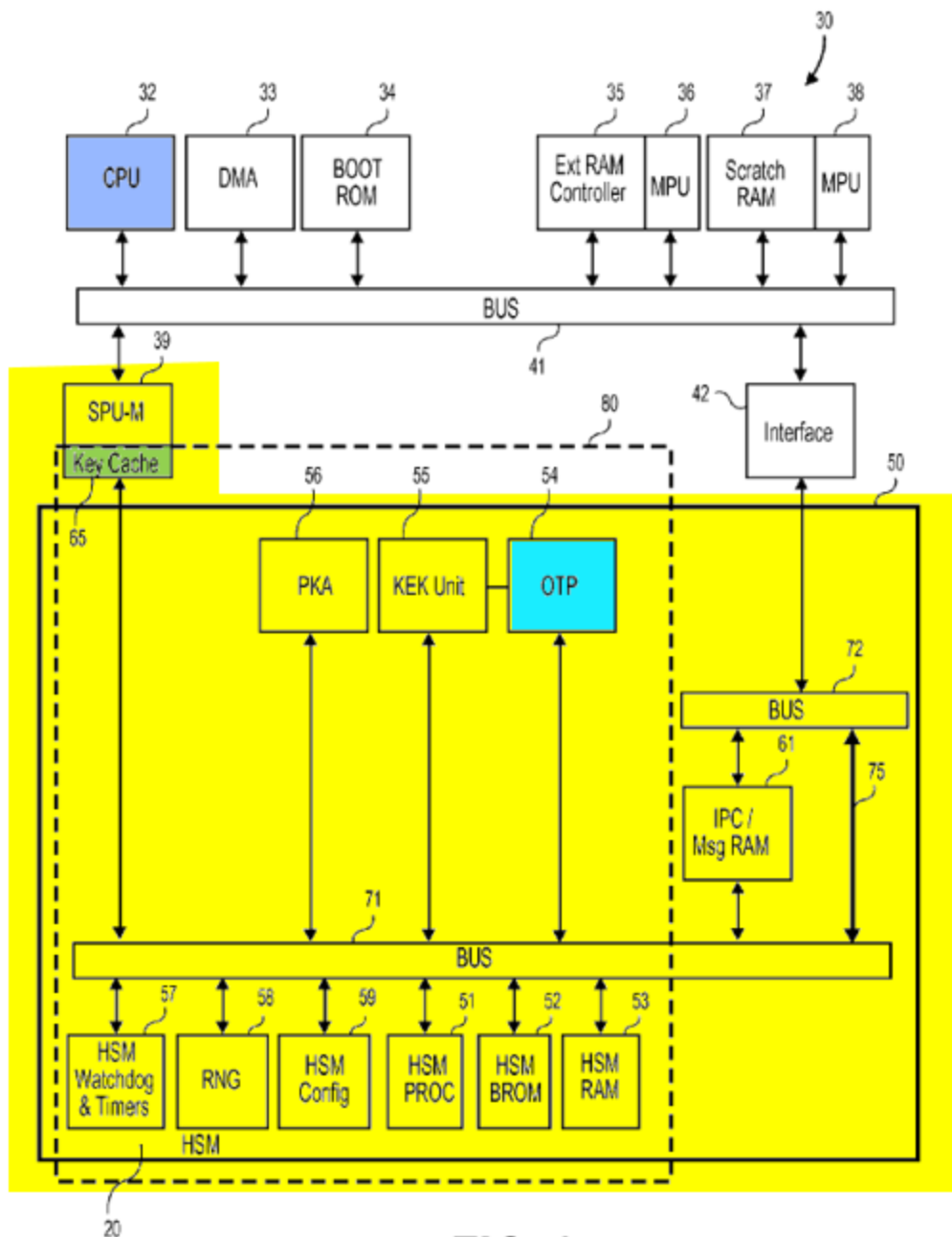


FIG. 3

Ex-1008, Fig. 3.

The security engine (HSM and SPU-M) has memory areas, including the OTP **54** (cyan), and Key Cache **65** (green), together forming the claimed “secure memory.” Ex-1002 ¶107-108. Each of these memory areas are accessible only by

the security engine (HSM and SPU-M). For example, Figure 3 shows both Key Cache **65** and OTP **54**, inside the dashed line (highlighted in red below), which “represents the actual secure domain boundary that is not accessible by non-secure components.” Ex-1008 ¶34. In other words, the Key Cache **65** and OTP **54** (i.e., the “secure memory”) are accessible only by the components within the dashed line (portions of HSM and SPU-M, together the claimed “security engine”). Ex-1002 ¶107-108; Ex-1008, Figs. 1, 3.

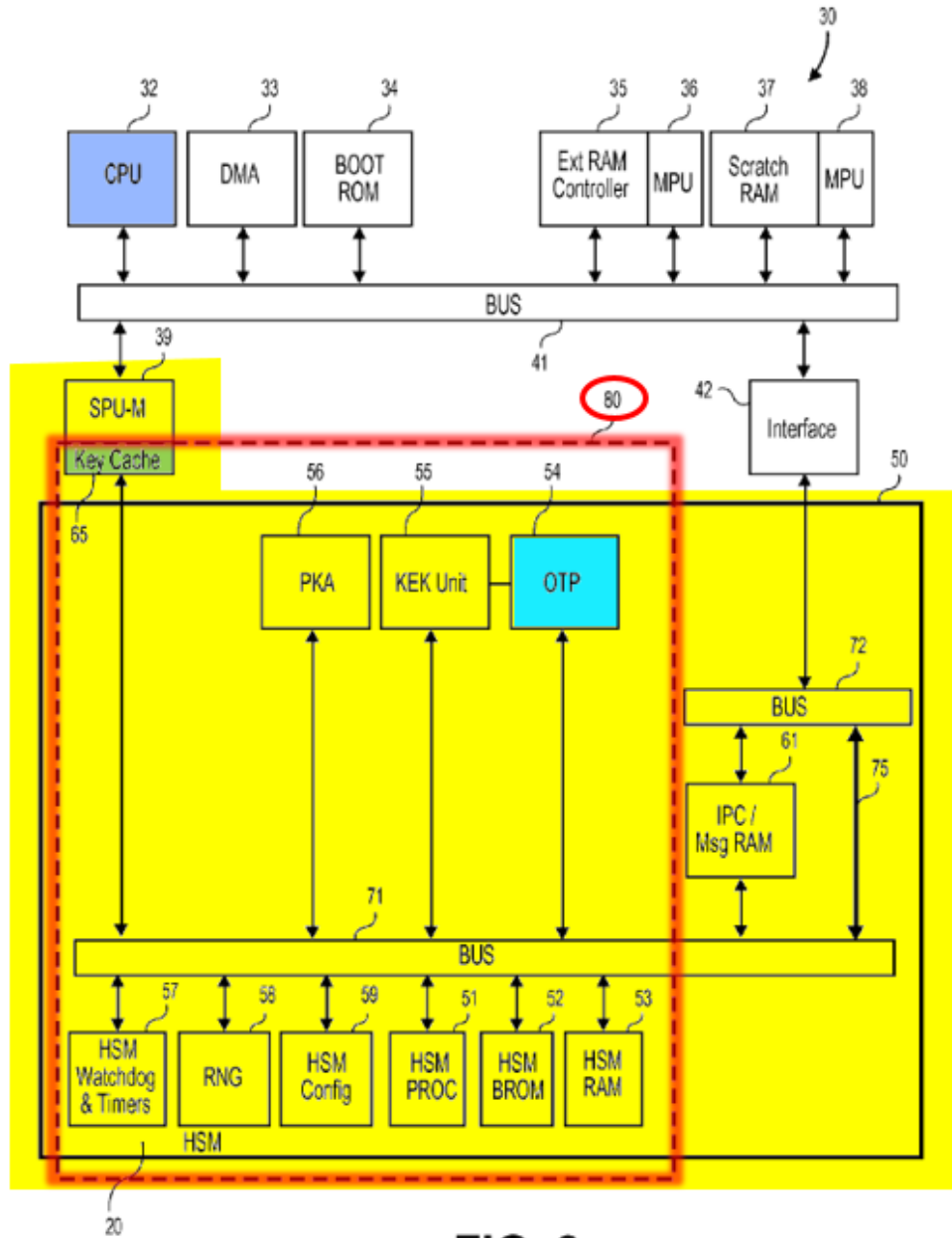


FIG. 3

Ex-1008, Fig. 3.

Chou's OPT 54 (part of the "secure memory") includes a "device unique key" programmed at the time of manufacturing of the SoC, and stored in OTP 54.

Ex-1008 ¶¶34, 38. The OTP (One Time Programmed memory) holds the "device unique key." Ex-1002 ¶109; Ex-1008 ¶38. Chou also explains that the OTP is

programmed during manufacturing. *Id.* Thus, a POSITA would have understood Chou to disclose that the OTP 54 would be programmed during manufacturing of the SoC, and in turn, understood the “device unique key,” stored in the OTP 54, would also be programmed during manufacturing of the SoC. Ex-1002 ¶110.

- c. **1[b] generate a random nonce for initiating a request for a secure communication session with a remote server over a network using the nonce;**

Chou’s HSM includes a random number generator (RNG) (highlighted pink). The RNG generates a true random number (i.e., a random nonce) used as a seed for cryptographic algorithms. Ex-1002 ¶111; Ex-1008 ¶34, 42. Thus, Chou discloses the random number would have been used during the cryptographic key exchange that requires a random seed. Ex-1002 ¶¶112-113.

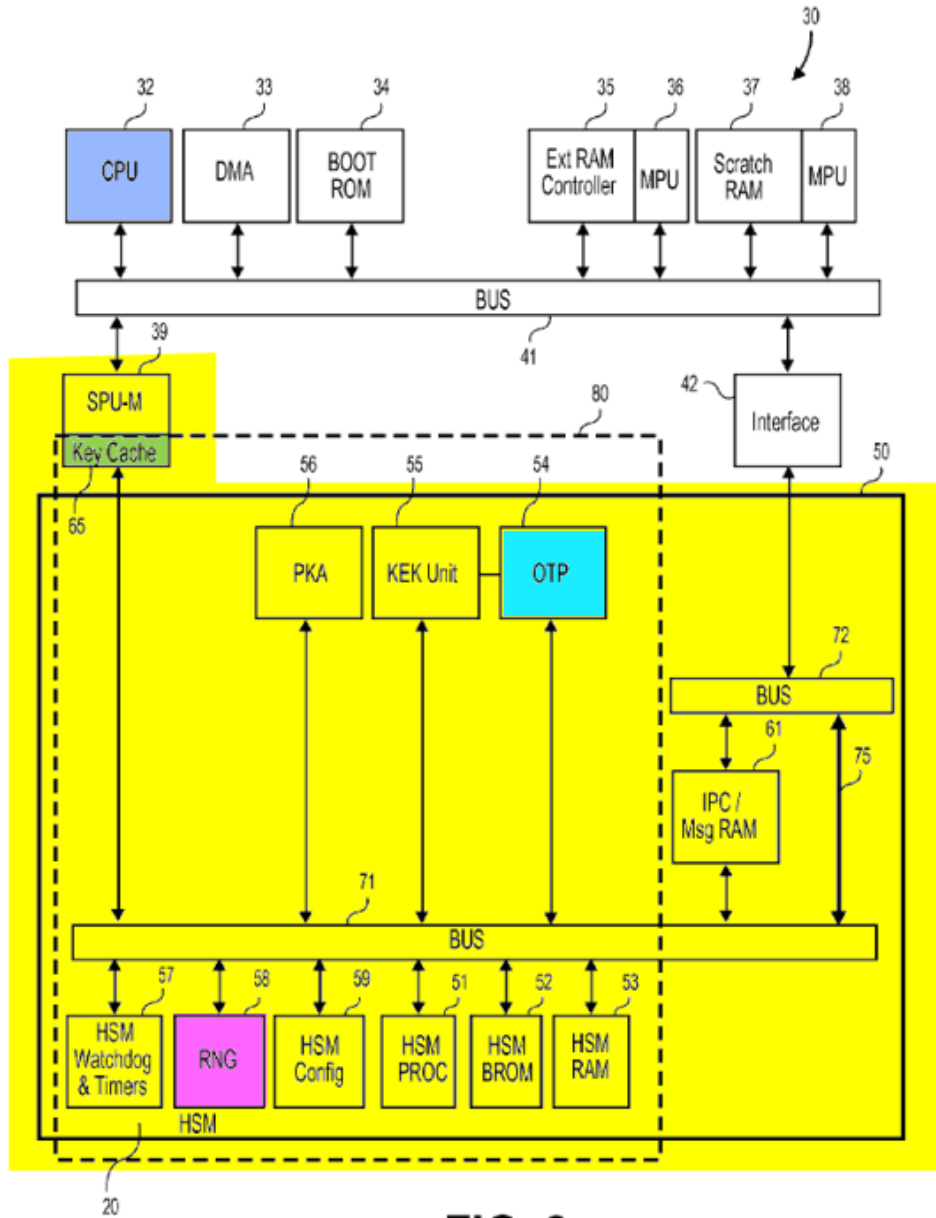


FIG. 3

Chou further discloses that the HSM establishes a secure communication session with a remote server using the nonce. Chou's invention is used to securely manage the various session keys, which include the keys used in the context of financial transactions with a server over a network, such that the keys are not revealed to unauthorized resources. Ex-1002 ¶114; Ex-1008 ¶6-9.

d. 1[c] perform a cryptographic key exchange with the remote server;

Chou's HSM includes a Public Key Accelerator (PKA) module (highlighted purple) that performs a number of symmetric key algorithms, including the "Diffie-Hellman public value and *shared key computation*" and the "elliptic curve Diffie-Hellman." Ex-1002 ¶115; Ex-1008 ¶43. Diffie-Hellman is a method of establishing a shared key between two separate entities (such as a client device and a remote server) without having to explicitly exchange the key itself. Ex-1002 ¶¶35-50. Thus, the PKA establishes symmetric keys such as the Diffie-Hellman shared key. *Id.* ¶115-117. Thus, Chou's HSM performs the cryptographic exchange with a remote server.

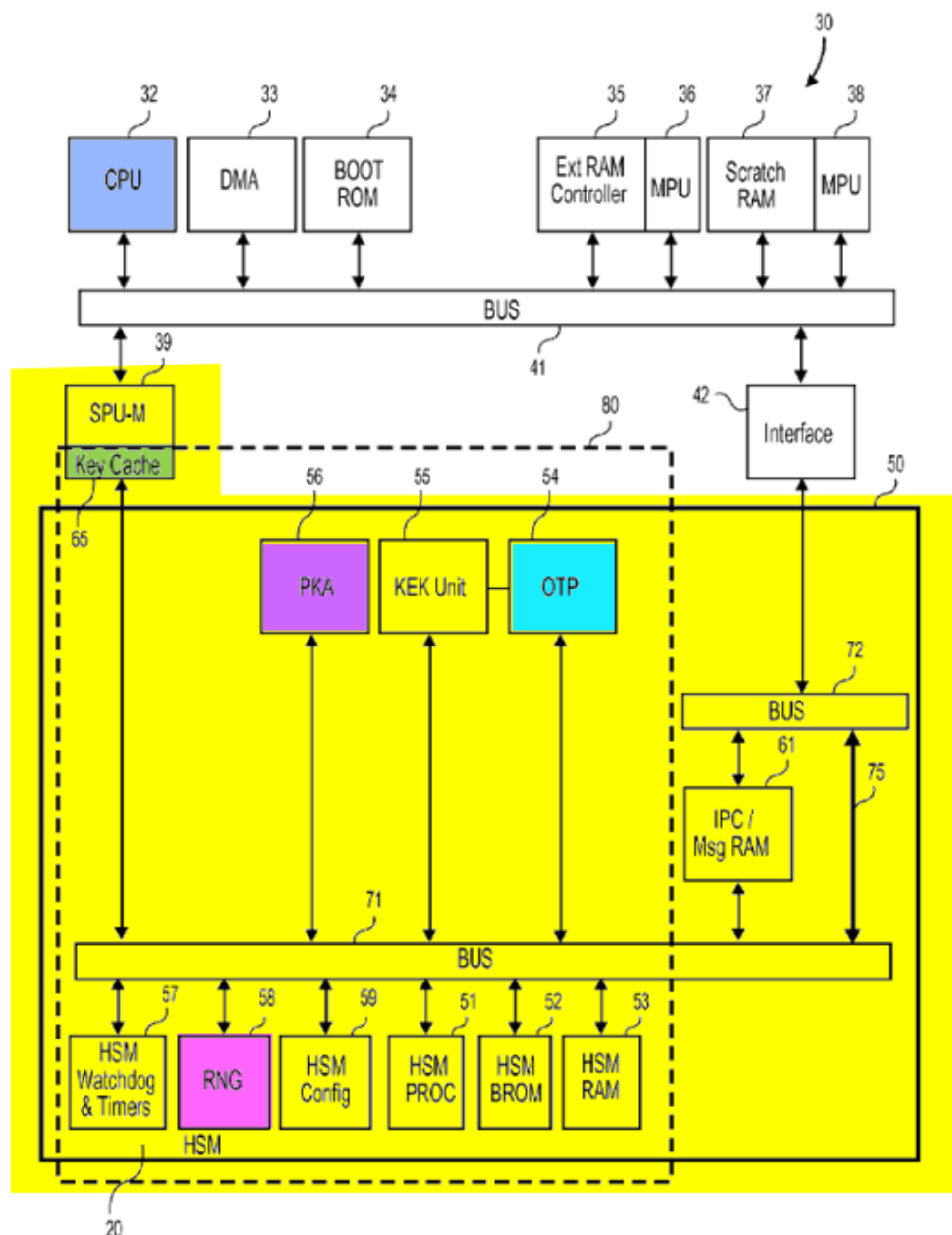


FIG. 3

- e. **1[d] 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;**

As explained above, the PKA establishes session keys using key exchange algorithms, such as Diffie-Hellman key exchange or Elliptic curve Diffie-Hellman

exchange. *See* Ex-1008 ¶43. As Dr. Crovella explained, Diffie-Hellman (or its variant, Elliptic curve Diffie-Hellman) is a key exchange algorithm that allows the two parties to separately generate a common (i.e., symmetric) session key for a secure message exchange. Ex-1002 ¶¶35-50. Thus, Chou discloses generating a symmetric session key based on the Diffie-Hellman exchange. *Id.* ¶118.

In addition, Chou's SPU-M (also part of the claimed "security engine") is "a hardware accelerator for *symmetric key algorithms*" that "implement[s] many *symmetric* key algorithms such as AES, DES, RC4, SHA1-HMAC, SHA2-HMAC, MD5-HMAC, as well as others." Ex-1008 ¶¶54-55. In other words, the SPU-M encrypts and decrypts data using the shared keys generated by PKA, using *symmetric* key algorithms. Ex-1002 ¶119.

Chou explains that the "secure key" (such as the Diffie-Hellman shared key computed by the PKA) that the HSM manages are used to encrypt and decrypt messages exchanged—i.e., they are "symmetric session keys" for the communication session between the device and the remote server. For example, Chou explains that, using the secure key mechanism, "a sender will encrypt the data that is to be transmitted" (Ex-1008 ¶5) and "the recipient of the secure data utilizes a secure key to decrypt the data" (¶6)—i.e., the "secure key" is used to encrypt and decrypt the data transmitted between two parties (e.g., in a secure

financial transactions), and therefore is a “symmetric session key” for the transaction session. Ex-1002 ¶¶120-121.

f. 1[e] encrypt the symmetric session key based on the security key; and

Chou’s HSM performs wrapping (i.e., encryption) of the secure keys (the claimed “symmetric session keys”). For example, the “secure key functions” performed by the HSM “includ[e] the wrapping and unwrapping of keys.” Ex-1008 ¶37; Ex-1002 ¶¶122-124. The “secure key” is wrapped/unwrapped using a “device unique key” (i.e., the claimed security key). For example, a “content key,” which is a type of a “secure key,” is unwrapped using the device unique key. Ex-1008 ¶25 (“unwrapping of the content key is performed strictly within HSM 20, such as by using the device unique key stored in secure key module 23.”); ¶69 (“Secure keys are wrapped/unwrapped and authenticated within the secure zone....”); Ex-1002 ¶124.

g. 1[f] store the encrypted session key in the secure memory,

Chou further discloses storing the data processed by the security processor, such as the session key wrapped by the HSM, in the secure memory. For example, Chou states that the “[s]ecure 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.” Ex-1008 ¶21; Ex-1002 ¶¶125-

126. Secure memory **22** (inside the HSM) “is utilized to store data that is processed and/or to be processed by security processor.” Ex-1008 ¶21; Ex-1002 ¶¶242. Figure 1 shows the secure memory **22** (cyan) and secure key module **23** (orange) inside the HSM.

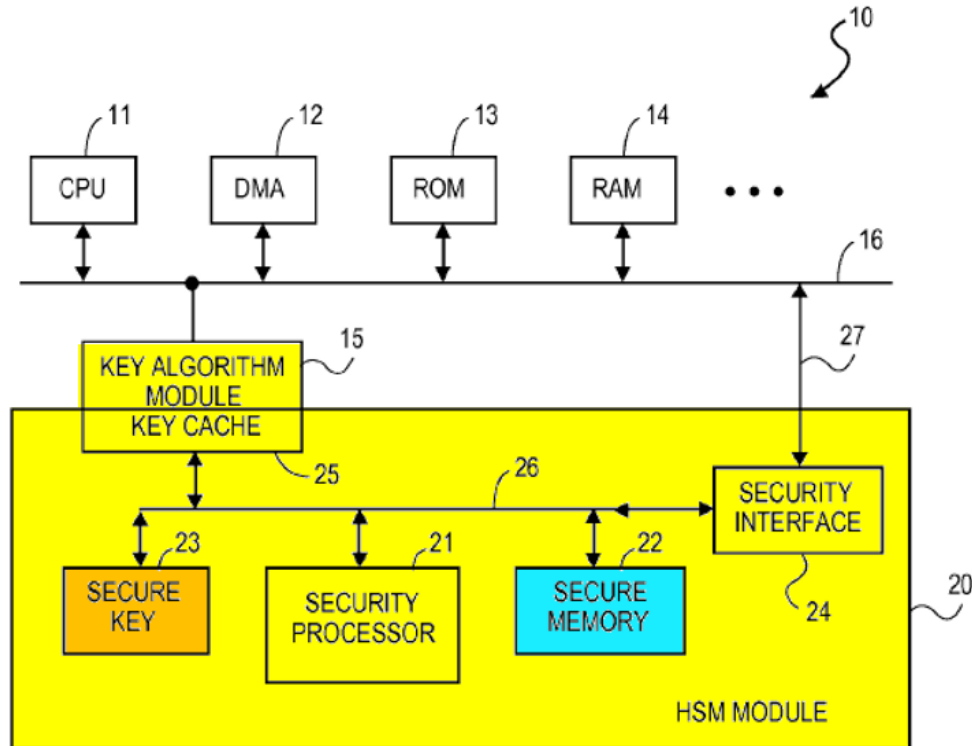


FIG. 1

Ex-1008, Fig. 1.

The encrypted session key is one piece of information processed by the security processor. As explained above, the “secure key” (the session key) is wrapped/unwrapped using a “device unique key” (i.e., the security key) by the HSM, inside the secure zone. See §XI.A.1.f; Ex-1002 ¶¶125-126. Thus, the

encrypted session key is also stored in secure memory **22**, the memory location that stores data processed by HSM. *Id.* ¶126.

In addition, Chou explains “[k]eys for [the symmetric key algorithms] may be loaded by HSM **20** into key cache **65**.” Ex-1008 ¶55. Because the “key cache” is a volatile memory, a POSITA would have understood that the keys would be stored elsewhere in the HSM, such as the secure key module **23** that stores the key values (such as the secure keys) used to encrypt and decrypt data. Ex-1008, ¶21; Ex-1002 ¶¶127-128. Thus, POSITA would understand that the symmetric session key would be stored in a storage area (such as the secure key module **23** or the secure memory **22**) and loaded into key cache for the SPU-M to use in symmetric encryption. Ex-1002 ¶128.

h. 1[g] the system-on-a-chip to establish the secure communication session with the remote server over the network using the session key.

Chou’s SoC establishes a secure communication session, such as financial transaction sessions, with a remote server. Ex-1008 ¶6; Ex-1002 ¶129. The transaction is protected using a “secure key.” *Id.* Chou provides a secure key management scheme to manage the handling of such secure keys. Thus, a POSITA would have understood that Chou’s HSM is designed to be used in SoCs that establish the secure communication session with a remote server over the network using the session key. *Id.*

2. Claim 24

- a. 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.**

Chou discloses a connection between the security engine (the HSM **20** and SPU-M **15**) and a CPU through a security interface that connects to a connection or bus (connection **27**, between the security interface and the connection to CPU).

The HSM **20** is communicatively coupled to the CPU through the connection that goes through the security interface **24**. Ex-1008 ¶24 (“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**.”); Ex-1002 ¶130. This interconnect is on the system-on-a-chip because the entire “device **10**, including HSM **20**, is constructed on a single integrated circuit chip.” Ex-1008 ¶20.

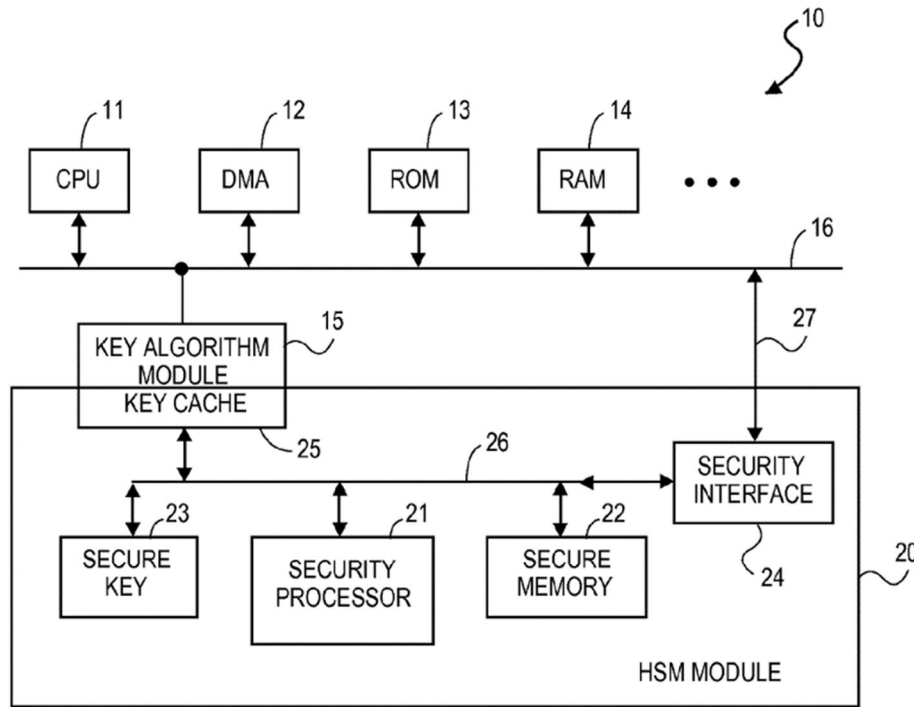


FIG. 1

3. Independent Claims 9, 17

a. 9[pre] A method comprising;

17[pre] 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:

Chou discloses a method. Ex-1008, Cover (“Apparatus and Method for Providing Hardware Security”); *see also*, ¶¶10.

Chou also explains the operations performed by its invention may embodied in processors executing appropriate software. Ex-1008 ¶71. A POSITA would recognize that such software would be stored in a non-transitory machine-readable

storage media comprising a plurality of instructions that are executed when the device executes. Ex-1002 ¶¶131-132.

- b. **9[a] 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;**

17[a] 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;

See §XI.A.1.b, 1[a]; §XI.A.1.c, 1[b]; Ex-1002 ¶¶133, 105, 111-114.

- c. **9[b] 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;**

17[b] initiate a request for a secure communication session with a remote server over a network, the request including the random nonce;

See §XI.A.1.c, 1[b]; Ex-1002 ¶¶134, 111-114.

- d. **9[c] performing a cryptographic key exchange , using the security engine of the system-on-a-chip, with the remote server;**

17[c] perform a cryptographic key exchange, using the security engine of the system-on-a-chip, with the remote server;

See §XI.A.1.d, 1[c]; Ex-1002 ¶¶135, 115-117.

- e. **9[d] 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;**

17[d] 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;

See §**XI.A.1.e**, 1[d]; Ex-1002 ¶¶136, 118-121.

- f. **9[e] 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;**

17[e] 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;

See §**XI.A.1.f**, limitation 1[e] (“encrypt the symmetric session key based on the security key”) and Section **XI.A.1.b**, limitation 1[a] (“...the secure memory includes a security key that was encoded in the secure memory during a manufacturing process of the system-on-a-chip”); Ex-1002 ¶137, 122-124, 107-110.

- g. **9[f] storing the encrypted session key in the secure memory of the security engine of the system-on-a-chip; and**

17[f] store the session key in the secure memory of the security engine of the system-on-a-chip; and

See §**XI.A.1.g**, 1[f]; Ex-1002 ¶¶138, 125-128.

- h. 9[g] establishing, using the client device, the secure communication session with the remote server using the session key.**

17[g] establish the secure communication session with the remote server using the session key.

See §XI.A.1.h, limitation 1[g]; Ex-1002 ¶¶139, 129.

B. Ground 2: Chou, in combination with TLS-1.0

To the extent Chou in combination with the knowledge of a POSITA (Ground 1) does not disclose all limitations of the challenged claims, those limitations are obvious in further combination with TLS-1.0, as explained below. The contemplated combination retains the structure of Chou for creating and managing the secure keys, wherein the secure key is created in accordance with the SSL/TLS protocol as specified by TLS-1.0.

1. Motivation to Combine

The SSL/TLS protocol specifies a standard protocol to be used by two communicating entities (such as a client device and a web server) to establish an encrypted and authenticated communication channel. Ex-1002 ¶¶61-70, 141. The SSL/TLS protocol was widely used by the priority date of the '838 Patent. *Id.* ¶¶141.

A POSITA would have recognized that the SSL/TLS protocol was a well-known method of establishing a secure communication transaction through an unsecure network, such as the internet. By the priority date of the '838 Patent,

TLS was a widely-accepted method of securing communications between two parties over the internet. Ex-1002 ¶143. TLS was widely supported by web browsers, servers, and other applications at the time, making it a standard protocol for secure communication on the internet. A POSITA would have realized that, by adopting the TLS standard, Chou’s system would be improved to allow it to communicate securely with a wide variety of servers. Thus, a POSITA would have been motivated to incorporate TLS-1.0 as the standard for creating session keys over the internet, in such scenarios where “secure data transfer is utilized to perform secure financial transactions over a network” as contemplated in Chou. Ex-1008 ¶¶6; Ex-1002 ¶143; *see also, Intel Corp. v. PACT XPP Schweiz AG*, No. 2022-1037, 2023 WL 2469631, at *5 (Fed. Cir. Mar. 13, 2023) (“There is a motivation to combine when a known technique ‘has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way.’”).

In addition, TLS-1.0 improves mutual authentication of the client and server, and data integrity. Ex-1002 ¶144. A POSITA would therefore have been motivated to incorporate TLS-1.0 into communication scenarios using Chou’s HSM device, and to use Chou’s HSM device to manage the TLS session key to further secure it. *Id.*

A POSITA would have been motivated to use TLS-1.0 to utilize the available SSL/TLS open source libraries, thereby saving significant amounts of development time, resources, and manpower. Ex-1002 ¶145.

A POSITA would have had a reasonable expectation of success in implementing TLS-1.0 because she could take advantage of the large community of developers who contribute to the code, provide support, and continuously improve open source libraries for SSL/TLS. Ex-1002 ¶145. Open-source libraries are also subject to peer review and well-tested to ensure quality and reliability. Thus, a POSITA could rely on the widely available open-source community resources and their experiences, and have a reasonable expectation of success. *Id.*

2. Dependent Claim 3

- a. 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 '838 Patent claims encrypting a pre-master key with the security key, but does not describe this subject matter in its specification. Instead, it describes the normal TLS operation, i.e., encrypting a pre-master key with the *server's public key*. Ex-1001, 8:28-32; 10:60-64; Ex-1002 ¶146. To the extent the specification describes Claim 3, “encrypt the pre-master key *using the security key of the security engine*,” Chou in combination with TLS-1.0 also describes this

element because TLS-1.0's disclosure matches the '838 specification. Ex-1002

¶¶147-149.

TLS-1.0 discloses an RSA-encrypted premaster secret message, which requires the client device (such as Chou's SoC device) to generate a pre-master key, encrypt it using the public key of the server, and send the result to the server:

7.4.7.1. RSA encrypted premaster secret message

Meaning of this message:

If RSA is being used for key agreement and authentication, the client generates a 48-byte premaster secret, encrypts it using the public key from the server's certificate or the temporary RSA key provided in a server key exchange message, and sends the result in an encrypted premaster secret message. This structure is a variant of the client key exchange message, not a message in itself.

Ex-1006, §7.4.7.1. Thus, TLS-1.0 discloses the same operation as the '838 Patent.

3. Dependent Claim 4

- a. 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 security engine is further to wrap the pre-master key using a server public key”: As explained in Section **XI.B.2** above, TLS-1.0 specifies that the premaster secret is encrypted using the server (responder)'s public key. Ex-1002

¶150.

“store the wrapped pre-master key in the secure memory”: Chou further discloses storing data to be processed by the security processor in secure memory. Ex-1002 ¶151. For example, Chou discloses a Hardware Security Module (HSM) that includes a memory used to store data processed and to be processed by the security processor. Ex-1008 ¶21 (“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**.”); Ex-1002 ¶151.

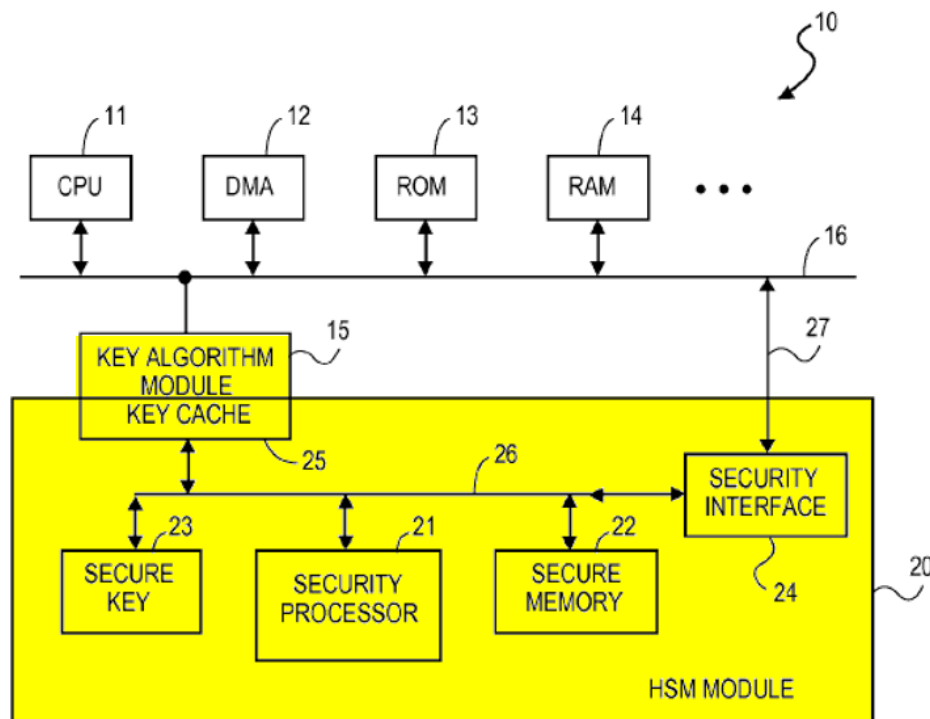


FIG. 1

Ex-1008, Fig. 1.

In the combination where Chou's system is used to generate a secure key according to TLS-1.0, the key exchange would be performed by security processor 21. Ex-1002 ¶152. Thus, the encrypted pre-master key (generated by HSM according to TLS-1.0), is one piece of data processed by the security processor inside the HSM, and therefore stored in secure memory in the HSM. *Id.* ¶¶152-153.

4. Dependent Claim 5

- a. **5[pre]/5[a]: 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**

TLS-1.0 specifies that the "master secret" (session key) is created by the following method:

master_secret = PRF(pre_master_secret, "master secret",
ClientHello.random + ServerHello.random) [0..47];

Ex-1006, §8.1. This equation specifies that the master secret (session key) is generated by applying the Pseudo-Random Function (PRF) to the pre-master secret (which is generated using the Diffie-Hellman, RSA, or other key exchange methods), along with the "master secret" label and a concatenated value of the client and server random values (ClientHello.random + ServerHello.random). Ex-1002 ¶¶154-156. This key generation method takes the pre_master_secret as input, i.e., generates the session key as a function of the pre-master key. *Id.* ¶7.

b. 5[b] calculate a hash function as a function of the random nonce generated by the security engine and a nonce received from the remote server

Chou discloses that the PKA module (part of the HSM) performs hash calculation. Ex-1008 ¶¶43-45 (“PKA module 56 is a specialized processor that provides acceleration for a number of algorithms” including “1024-bit DSA signature with SHA-1 hash or 2048-bit DSA signature with SHA-256 hash.”). Ex-1002 ¶158. Chou does not explicitly state that the hash function is calculated as a function of the random nonce generated by the security engine and a nonce received from the remote server. *Id.* ¶¶159-160. This limitation, however, is disclosed in TLS-1.0.

The session key (master secret) creation in TLS-1.0, explained above, is:

master_secret = PRF(pre_master_secret, “master secret”,
ClientHello.random + ServerHello.random) [0..47];

Ex-1006, §8.1. The PRF (pseudo-random function) is made up of two hash functions, MD5 and SHA-1:

The PRF is then defined as the result of mixing the two pseudorandom streams by exclusive-or'ing them together.

PRF(secret, label, seed) = P_MD5(S1, label + seed) XOR
P_SHA-1(S2, label + seed);

Ex-1006, §5; *see also, id.*, §B (glossaries for MD5 and SHA showing they are hashing functions); Ex-1002 ¶161.

Thus, during the key exchange according to TLS-1.0, the session key is generated as a result of calculating PRF (which includes calculating MD5 and SHA hash functions) using, among others, the “ClientHello_random” (i.e., the random nonce generated by the security engine) and the “ServerHello_random” (i.e., a nonce received from the remote server) as two of the inputs. In other words, the client calculates a hash function as a function of the two random nonces. Ex-1002 ¶162.

5. Dependent Claim 6

- a. **6[pre]/6[a]. 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 “private Diffie-Hellman client key” is a random number x_i , initially generated by the initiator (the client) at the beginning of the Diffie-Hellman algorithm (and kept private). Ex-1002 ¶¶163. The “public Diffie-Hellman client key” is the value $y_i = g^{x_i} \bmod p$, generated using the private Diffie-Hellman client key (x_i). This value y_i is sent to the responder via an unsecured channel, and is therefore “public.” Ex-1002 ¶163; Ex-1005. Chou’s PKA performs the “Diffie-Hellman public value and shared key computation of modulus sizes up to 4096 bits.” Ex-1008 ¶¶43-44; Ex-1002 ¶164. TLS-1.0 also discloses a client exchange key message that establishes a shared Diffie-Hellman premaster secret. Ex-1002 ¶¶165-166. The Diffie-Hellman key exchange algorithm must necessarily generate

a “private Diffie-Hellman client key” (x_i) to create the “public Diffie-Hellman client key” (y_i). *Id.* Thus a POSITA would have understood that Chou’s HSM generates a public Diffie-Hellman client key and a private Diffie-Hellman client key. *Id.* ¶166.

b. 6[b] sign the public Diffie-Hellman client key using the security key of the security engine,

TLS-1.0 further explains the specific form of the client key exchange message used when the Diffie-Hellman key exchange method is chosen to establish the session’s secure key. This message includes the public Diffie-Hellman client key. Ex-1002 ¶¶167-168. In particular, a TLS-1.0 message “Client Diffie-Hellman public value” conveys the client’s Diffie-Hellman public value. This message is later signed using the security key and sent to the server:

7.4.7.2. Client Diffie-Hellman public value

Meaning of this message:

This structure conveys the client’s Diffie-Hellman public value (Y_c) if it was not already included in the client’s certificate.

Ex-1006, §7.4.7.2; Ex-1002 ¶169. This variant of the ClientKeyExchange message contains the client’s Diffie-Hellman public value. Ex-1002 ¶170.

TLS-1.0 Section 7.4.8 further describes a “Certificate Verify” message. This message is used by the client to send a digital signature over the entire handshake message up to this point (including the ClientKeyExchange message). Ex-1002

¶¶71-76, 171. The “Signature” is generated by first computing the hash value of all messages exchanged between the client and the server, and signing the value using the private key corresponding to the client’s certificate. *Id.* ¶¶172-174. The resulting signature is then included in the “Certificate Verify” message and sent to the server. *Id.* ¶174. Thus, TLS-1.0 discloses signing “Client Diffie-Hellman public value” message, which includes the public Diffie-Hellman client key, using a device-specific key of the client, such as the “device unique key” of Chou.

c. 6[c] send the signed public Diffie-Hellman key to the remote server.

The “Certificate Verify” message (which includes the signed public Diffie-Hellman client key) is sent from the client to the server in TLS-1.0. Ex-1002 ¶175.

6. Dependent Claim 7

a. 7[a] 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

Chou discloses the PKA performing the Diffie-Hellman shared key computation. Ex-1008 ¶¶43-53. To compute Diffie-Hellman shared key, the PKA must receive the public Diffie-Hellman server key (y_r), such that it can compute the Diffie-Hellman shared key from the public Diffie-Hellman server key and the private Diffie-Hellman client key. Ex-1002 ¶¶35-50, 177; Ex-1005.

- b. 7[b] generate the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key.**

Chou's PKA (in the HSM) computes the Diffie-Hellman shared key (Ex-1008, §43-44), from the public Diffie-Hellman server key and the private Diffie-Hellman client key. Ex-1002 ¶178. In TLS-1.0, this Diffie-Hellman shared key is the "pre-master key." *Id.* ¶179. The "master_secret" (the session key) is created by the following method:

master_secret = PRF(pre_master_secret, "master secret",
ClientHello.random + ServerHello.random) [0..47];

Ex-1006, §8.1.

One of the inputs to generate the master_secret is the pre_master_secret. Ex-1002 ¶¶180-181. The pre_master_secret (Diffie-Hellman shared secret) is, in turn, generated using the private Diffie-Hellman client key and the public Diffie-Hellman server key. Ex-1002 ¶¶35-50, 182; Ex-1005. Thus, the session key is generated using the private Diffie-Hellman client key and the public Diffie-Hellman server key. Ex-1002 ¶182.

7. Dependent Claims 10, 18

- a. **10[pre]. 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 further comprising:**

18[pre]. 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, including the nonce, to the remote server; and

TLS-1.0 specifies the types and sequences of the messages exchanged between the client and the server to establish a secure session. The initial message sent from the client to the server is “ClientHello”:

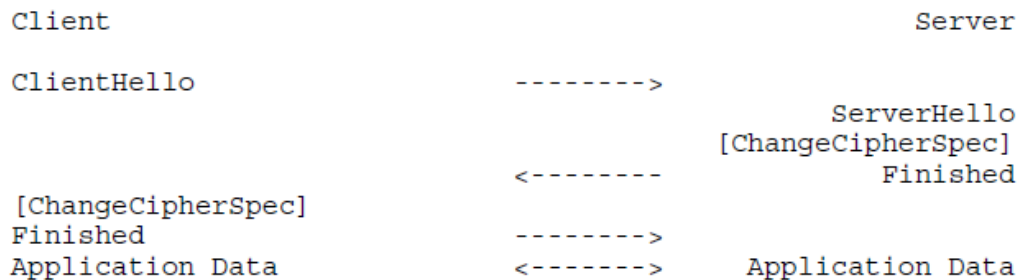


Fig. 2 - Message flow for an abbreviated handshake

Ex-1006, §7.3, Fig. 2. The ClientHello message includes ClientHello.random, which is a random number generated by the client. Ex-1002 ¶¶184-186.

7.4.1.2. Client hello

...

Structure of this message:

The client hello message includes a random structure, which is used later in the protocol.

```
struct {  
    uint32 gmt_unix_time;  
    opaque random_bytes[28];  
} Random;  
  
...  
random_bytes  
  
    28 bytes generated by a secure random number  
    generator.
```

Ex-1006, §7.4.1.2.

- b. 10[a] receiving a ServerHello message from the remote server, the ServerHello message including a server nonce; and**

18[a] 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

The next TLS-1.0 message sent in response to the ClientHello is the ServerHello message from the server to the client. Ex-1002 ¶¶187-188.

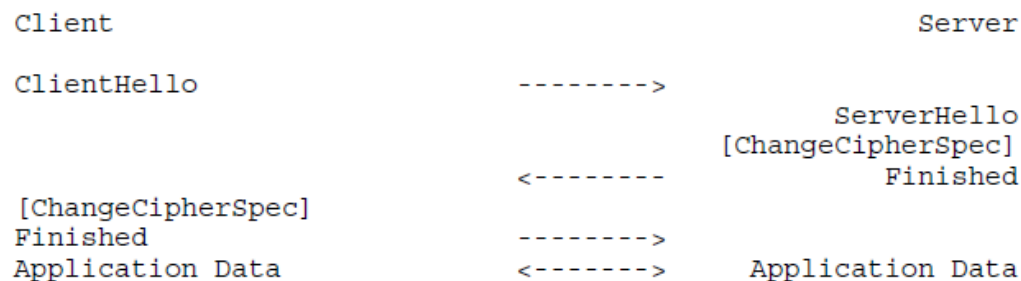


Fig. 2 - Message flow for an abbreviated handshake

Ex-1006, §7.3, Fig. 2. The ServerHello message includes ServerHello.random, which is “generated by the server and must be different from (and independent of) ClientHello.random.” Ex-1006, §7.4.1.3; Ex-1002 ¶189.

- c. **10[b] storing the server nonce in the secure memory of the security engine of the system-on-a-chip.**

18[b] store the server nonce in the secure memory of the security engine of the system-on-a-chip.

Chou discloses storing the data to be processed by the security processor in the secure memory. Ex-1002 ¶190. For example, Chou explains that the HSM includes a secure memory used to store data processed and/or to be processed by the security processor. Ex-1008 ¶21; Ex-1002 ¶190. In the combination where Chou's system is used to generate a secure key according to TLS-1.0, the key exchange would be performed by the security processor. Ex-1002 ¶191. The server nonce (the "ServerHello.random") is one piece of data the PKA module uses to generate the session key, and therefore is stored in the secure memory. *Id.* ¶192.

8. Dependent Claims 11, 19

- a. **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 of the security engine; and sending the encrypted pre-master key to the remote server.**

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 send the encrypted pre-master key to the remote server.

See §XI.B.2, Cl. 3; Ex-1002 ¶¶194, 146-149 (Cl. 3).

9. Dependent Claims 12, 20

- a. **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.**

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.

See §**XI.B.3**, Cl. 4; Ex-1002 ¶¶195, 150-153 (Cl. 4).

10. Dependent Claims 13, 21

- a. **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 system-on-a-chip as a function of the pre-master key.**

21. The one or more non-transitory machine-readable storage media of claim 19, wherein 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.

See §**XI.B.4.a**, 5[pre]/5[a]; Ex-1002 ¶¶196, 154-157 (5[pre]/5[a]).

11. Dependent Claims 14, 22

- a. **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.**
- 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.**

See §**XI.B.4.b**, 5[b]; Ex-1002 ¶¶197, 158-162 (5[b]).

12. Claims 15, 23

- a. **15[pre]/15[a] 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-chip, a public Diffie-Hellman client key and a private Diffie-Hellman client key;**
- 23[pre]/23[a] 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: sign the public Diffie-Hellman client key using the security key of the security engine**

See §**XI.B.5.a**, 6[a]; Ex-1002 ¶¶198, 163-166 (6[a]).

- b. 15[b] signing the public Diffie-Hellman client key using the security key of the security engine of the system-on-a-chip;**

23[b] sign the public Diffie-Hellman client key using the security key of the security engine of the system-on-a-chip;

See §**XI.B.5.b**, 6[b]; Ex-1002 ¶¶199, 167-174 (6[b]).

- c. 15[c] sending the signed public Diffie-Hellman key to the remote server;**

23[c] send the signed public Diffie-Hellman key to the remote server;

See §**XI.B.5.c**, 6[c]; Ex-1002 ¶¶200, 175 (6[c]).

- d. 15[d] receiving a public Diffie-Hellman server key from the remote server; and**

23[d] receive a public Diffie-Hellman server key from the remote server; and

See §**XI.B.6.a**, 7[a]; Ex-1002 ¶¶201, 177 (7[a]).

- e. 15[e] 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.**

23[e] 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.

See §**XI.B.6.b**, 7[b]; Ex-1002 ¶¶202, 178-182 (7[b]).

C. Grounds 3 and 4: Buer in view of knowledge of a POSITA or in view of Chou (Claims 1, 3-7, 9-15, 17-24)

To the extent Buer in view of the knowledge of a POSITA (Ground 3) does not disclose all limitations of the challenged claims, those limitations are obvious in further combination with Chou, as further explained below. The combination would retain the same structure of Buer, but further incorporate Chou's secure memory **22** (inside HSM **20**) as the storage area inside Buer's security module to store data processed or to be processed by the processor of the security module.

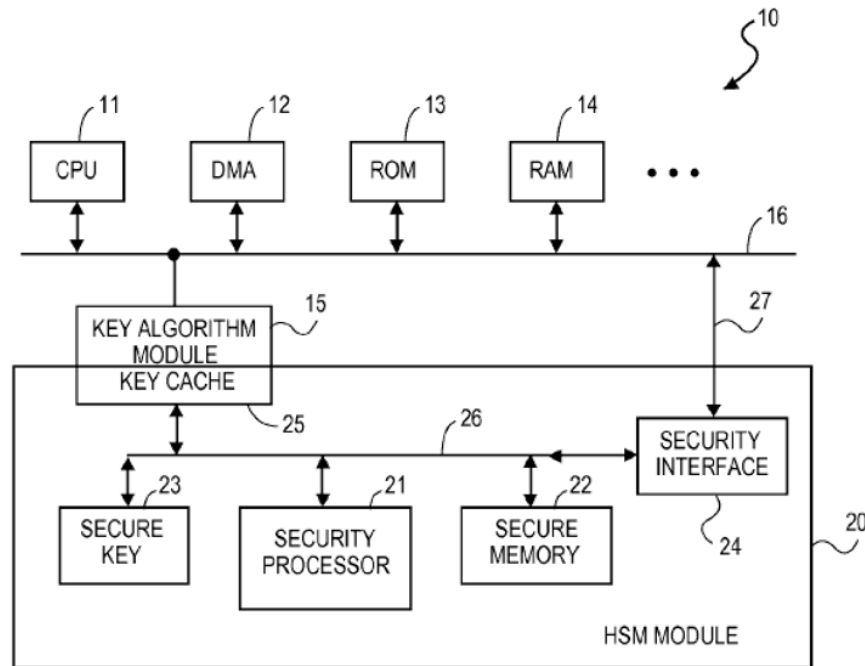


FIG. 1

Ex-1008, Fig. 1.

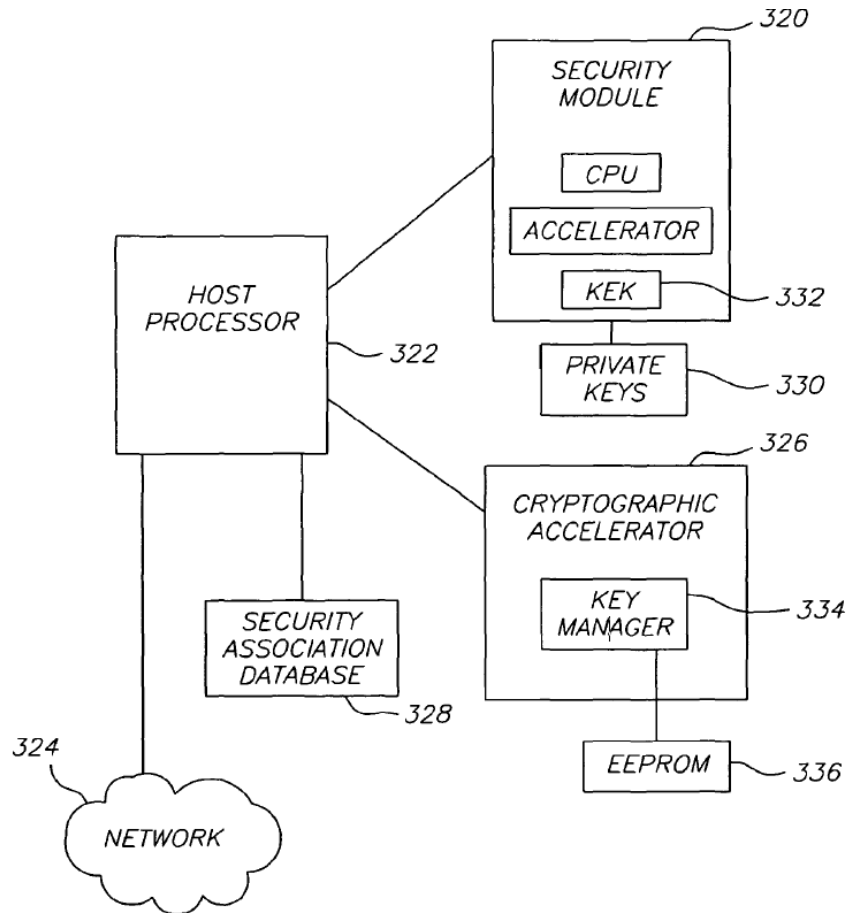


FIG. 3

Ex-1009, Fig. 3.

1. Motivation to Combine Buer with Chou

A POSITA would have recognized that Chou's secure memory inside the HSM provides several benefits. For example, by integrating a secure memory inside a security module, the system can provide a higher level of security for sensitive data. Ex-1002 ¶205. A POSITA would have also recognized that, by storing the data processed by Buer's security module inside the security module itself (rather than in a separate database outside the security module), overall

system performance is improved. Ex-1002 ¶206. By integrating the secure memory of Chou into Buer's security module, Buer's security module could further simplify the key management process. *Id.* ¶207. Thus, a POSITA would be motivated to incorporate Chou's secure memory into Buer's security module for these benefits. *Id.* ¶208.

In addition, would have an expectation of success for including storage within a security module, which was a well-known approach for securing sensitive data. Ex-1002 ¶209. A POSITA would have been familiar with the basic concepts of hardware-based encryption and decryption, as well as key management using a secure memory inside a security module. *Id.* Furthermore, adding internal memory is a logical extension of Buer's system, which already has non-volatile memory to store a Key Encryption Key (KEK) in a protected portion of the system. Ex-1009, 4:45-60 ("Alternatively, the non-volatile memory **134** may be integrated into the key manager's integrated circuit to control access to the non-volatile memory **134**."); *see also* Fig. 3; Ex-1002 ¶210. Expanding the non-volatile memory to further store the data processed by the security module would have eliminated the need to separately maintain and manage the security association database outside the security module, further simplifying the design of Buer's device and providing efficiencies. *Id.* ¶211. A POSITA would have realized that these changes would involve existing technology and would not require any

significant innovation. *Id.* Thus, a POSITA would have a reasonable expectation of success in combining Chou’s secure memory into Buer’s security module to practice a well-established data storage method. *Id.*

Thus, a POSITA would find it obvious to combine Chou’s secure memory into Buer’s security module because it is a logical extension of existing technology disclosed in Buer, and is a well-established practice to improve security, efficiency, and manageability of the key management system. Ex-1002 ¶212; *see also, Intel v. PACT*, 2023 WL 2469631, at *5.

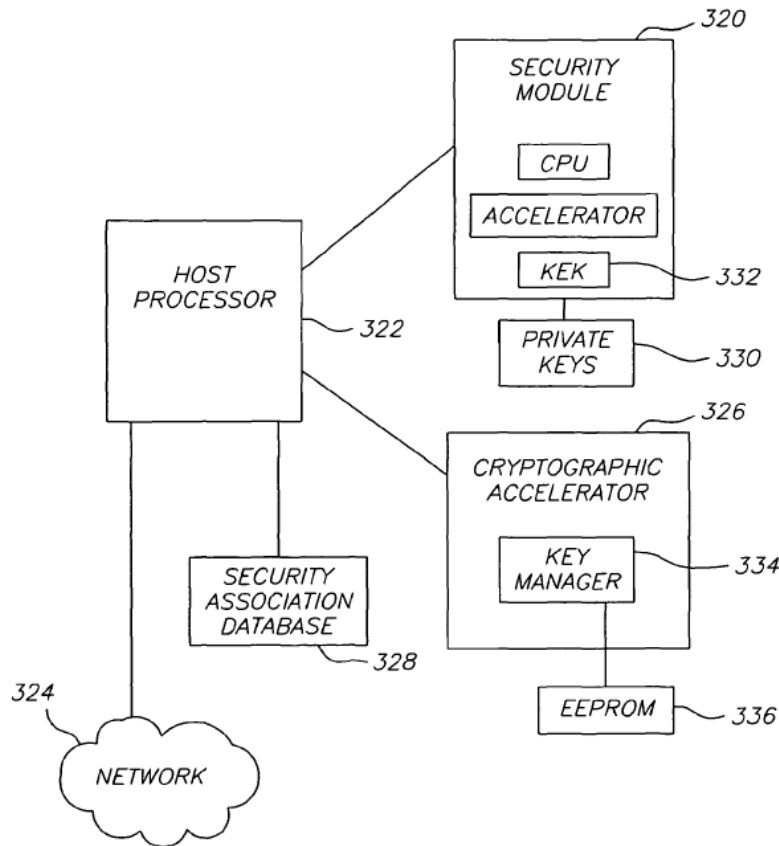
2. Independent Claim 1

a. 1[pre]

Buer discloses multiple processors and hardware modules (a host processor, security module, and cryptographic accelerator), and states that these can be combined in one integrated circuit, such as an SoC. Ex-1009, 16:36-46; Ex-1002 ¶213.

b. 1[a]

Buer’s device includes a “security module” and a “cryptographic processor” (together, the claimed “security engine”) that are separate from the host processor of the system-on-a-chip. For example, Buer’s Figure 3, reproduced below, shows Security Module **320** and Cryptographic Accelerator **326**, separate from Host Processor **322**. Ex-1002 ¶214.



Ex-1009, Fig. 3.

Buer's security engine (the security module and cryptographic accelerator) also includes a secure memory accessible only by the security engine. Ex-1002 ¶215. For example, Buer explains that its system includes a non-volatile memory, such as an EEPROM, accessible only by the key manager, which is part of the cryptographic accelerator, which is, in turn, part of the claimed "security engine." Ex-1009, 4:45-60; Ex-1002 ¶215.

The secure memory, such as non-volatile memory **134**, holds the Key Encryption Key (KEK), which is generated at the time of manufacturing the SoC.

See Ex-1009, 8:1-8 (“when the system is being manufactured the...KEK is loaded into the EEPROM....”). Ex-1002 ¶216.

c. **1[b]**

Buer discloses methods for establishing a secure channel between a main security module and a remote satellite security module (a remote server) through a network, as shown in Figure 13, reproduced below. The methods include a step of generating a random nonce. Ex-1002 ¶217.

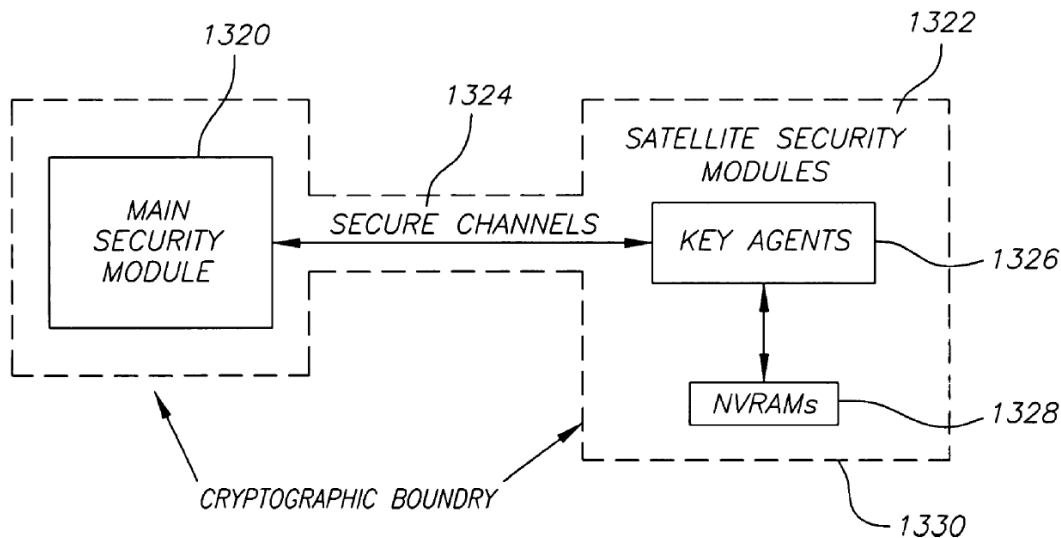


FIG. 13

Ex-1009, Fig. 13; 12:4-12 (“In a typical configuration, the satellite security modules **1322** will be located throughout a data network (e.g., the Internet).”); Ex-1002 ¶217.

To establish the secure channel, “the main security module **1320** initiates the secure channel exchanges.” Ex-1009, 13:1-4. Buer’s “security module” (part of

the claimed “security engine”), rather than the host processor of the client device, performs the various functions described in the secure channel exchange. Ex-1002 ¶219; Ex-1009, 13:1-7 (“[T]he main security module **1320** will create a message and send it to the host processor. The host processor then sends a command to the satellite security module **1322**. This command is followed by a response from the satellite security module **1322** [i.e., the remote server] that the host processor returns to the main security module **1320**.”).

Buer discloses the various cryptographic key exchange protocols, including “Unauthenticated Diffie-Hellman, One-Way RSA Authenticated Diffie-Hellman, Mutually Authenticated Diffie-Hellman and Shared Secret ReKey,” that can be used to establish the secure channel between the main security module and the satellite security module. Ex-1009, 13:1-20; Ex-1002 ¶220. Each of these protocols involves generating a random nonce for initiating the secure communication request. Ex-1002 ¶¶221-222. For example, Buer discloses, in the context of an Unauthenticated Diffie-Hellman exchange, that the initiator (i.e., the main security module) generates random number x_i , to create a public number, and sends it the responder (the satellite security module) with a nonce N_i in the initial steps of the exchange. Ex-1009, 13:21-44; *see also* 13:45-14:10 (One-Way RSA Authenticated Diffie-Hellman beginning with “Initiator generat[ing] random number x_i ” and “send[ing] y_i to the responder, along with a nonce N_i ”); 14:11-42

(Mutually Authenticated Diffie-Hellman); 14:43-59 (Shared Secret ReKey).

d. 1[c]

As explained above in Section **XI.C.2.c**, Buer discloses the main security module (part of the claimed “security engine”) performing various cryptographic key exchanges with the satellite security module (the claimed “remote server”). *See* Ex-1009, 13:21-44 (describing Unauthenticated Diffie-Hellman key exchange protocol); 13:45-14:10 (One-Way RSA Authenticated Diffie-Hellman exchange); 14:11-42 (Mutually Authenticated Diffie-Hellman); 14:43-59 (Shared Secret ReKey); Ex-1002 ¶223.

e. 1[d]

The various cryptographic key exchange protocols disclosed in Buer lead to the initiator (the main security module) generating a shared key material. Ex-1002 ¶224. The “shared” key material is shared (i.e., common) between the main security module and the satellite security module (i.e., the claimed “remote server”), and can be used to encrypt and decrypt messages exchanged between the main and satellite security modules. *Id.* ¶224.

For example, Buer discloses, in the context of an Unauthenticated Diffie-Hellman exchange, the initiator (i.e., the main security module) generates shared key material based on various information exchanged between the main security module and the satellite security module. Ex-1009, 13:21-44. Ex-1002 ¶¶225-226.

A POSITA would have understood that the Diffie-Hellman exchange described in Buer would result in the same session key on both sides of the communication for encryption and decryption, i.e., a symmetric session key. The Diffie-Hellman key exchange algorithm is designed for each side to be able to securely generate a common encryption key based on information exchanged via an insecure channel. Ex-1002 ¶¶35-50. A POSITA would have understood that the “shared key material” independently generated on both sides of the key exchange disclosed in Buer can then be used either directly as a symmetric encryption/decryption key (i.e., symmetric session key) or to generate a symmetric encryption key according to the agreed-upon algorithm. Thus, a POSITA would have understood Buer to disclose the Initiator (the main security engine) to generate a symmetric session key based on the various disclosed key exchange protocols. *Id.* ¶227.

Buer’s main security module may be a part of a device that also includes a separate host processor, through which the communication with the satellite security module is made. *See* Ex-1009, 12:62-64 (“the secure channel **1324** may be established through the host processor (e.g., processor **1024** in FIG. 10).”); Ex-1002 ¶¶228-229. The host processor coupled to the main security module would cooperate with the host processor coupled to the satellite security module to “establish a set of keys for each session.” Ex-1009, 5:19-21; Ex-1002 ¶229. These

keys, such as the key established between the main security module and the satellite security module are “uniquely associated with a session and are referred to as session keys.” Ex-1009, 5:22-23. These session keys are symmetric because “each host processor may decrypt the data encrypted by the other host processor” using these keys. Ex-1009, 5:20-22. Ex-1002 ¶229.

f. 1[e]

Once the security module generates the symmetric session key, the session key is encrypted based on the Key Encryption Key (KEK, the claimed “security key”). Ex-1009, 5:43-51 (“In accordance with this embodiment of the invention, these session keys are encrypted using KEK **238** and stored in a database **240** as security associations.”); Ex-1002 ¶¶230-231. Buer further clarifies that the security module (the claimed security engine), rather than the host processor, encrypts the session key. Ex-1009, 6:41-43 (“the security module **320** encrypts them with a KEK **332** before they are sent to the cryptographic accelerator **326** (via the host processor **322**).”); *Id.*, 1:9-12 (“[t]o protect [the] private keys [such as the SSL session keys], most of the encryption and decryption operations are performed inside the security module.”). Ex-1002 ¶¶232-233.

g. 1[f]

(1) Disclosure in Buer (Ground 3)

Buer discloses storing the encrypted session key in a database. Ex-1009, 5:43-51 (“In accordance with this embodiment of the invention, these session keys

are encrypted using KEK **238** and stored in a database **240** as security associations.”); Ex-1002 ¶235. Buer does not restrict the security association database to be stored in a secure memory, but acknowledges that “the encrypted security associations may be stored in any data memory.” Ex-1009, 5:66-6:1. This would include, for example, non-volatile memory **134** connected to key manager **132** (see Fig. 1) or private keys store **330** attached to security module **320** (see Fig. 3). Ex-1002 ¶235.

A POSITA would have recognized that, because Buer contemplates encrypting and decrypting messages using the security module that communicates with the cryptographic accelerator, it would be most efficient to store the session keys within the cryptographic boundaries of the security module or the cryptographic accelerator, thus eliminating the need for additional communication with an otherwise unsecured host processor. Ex-1002 ¶236. This can also improve system performance by reducing the number of memory accesses required to retrieve the encrypted session key. *Id.* Thus, A POSITA would have been motivated to store the encrypted session keys inside the secure memory directly attached to the security module. *Id.* In addition, a POSITA would have further recognized that, storing the security associations in the already-available non-volatile memory attached to the security module would eliminate the need for a separate storage area for the security association database, and would therefore

simplify the design of the system. Because the non-volatile memory is already access controlled by being coupled to the security engine, separate management or access control to the security association database is not needed. *Id.* ¶237.

Although Buer states that “database **240**” does not need to be protected, Buer does not teach away from storing the security associations in the secure memory. Ex-1002 ¶238. In fact, Buer states that “the encrypted security associations may be stored in *any* data memory.” Buer also explains that non-volatile memory can be made more secure by integrating into the same circuit as the key manager. Ex-1009, 17:1-4. Thus, Buer merely presents alternate design options to store the encrypted session key either in a more secure, but higher-cost location, or in a less secure, but more lower-cost location. Ex-1002 ¶238. A POSITA would have found either of these options possible, and would not have avoided storing the encrypted session keys in the secure memory. *Id.*

(2) Combination with Chou (Ground 4)

To the extent this limitation is not already disclosed in or rendered obvious by Buer as discussed above, Buer in combination with Chou (Ground 4) renders this limitation obvious.

As explained in Section **XI.A.1.g** (Ground 1, limitation 1[f]), Chou discloses a hardware security module (HSM) similar to Buer’s security module. Chou’s HSM further includes a secure memory (highlighted blue), which stores the data

processed by the HSM, the equivalent of Buer's CPU inside the security module.

Chou further discloses storing data processed by the security processor, such as the session key wrapped by the HSM, in the secure memory. Ex-1008 ¶21 (“Secure memory **22** is utilized to store data that is processed and/or to be processed by security processor **21**.”); Ex-1002 ¶¶241-243.

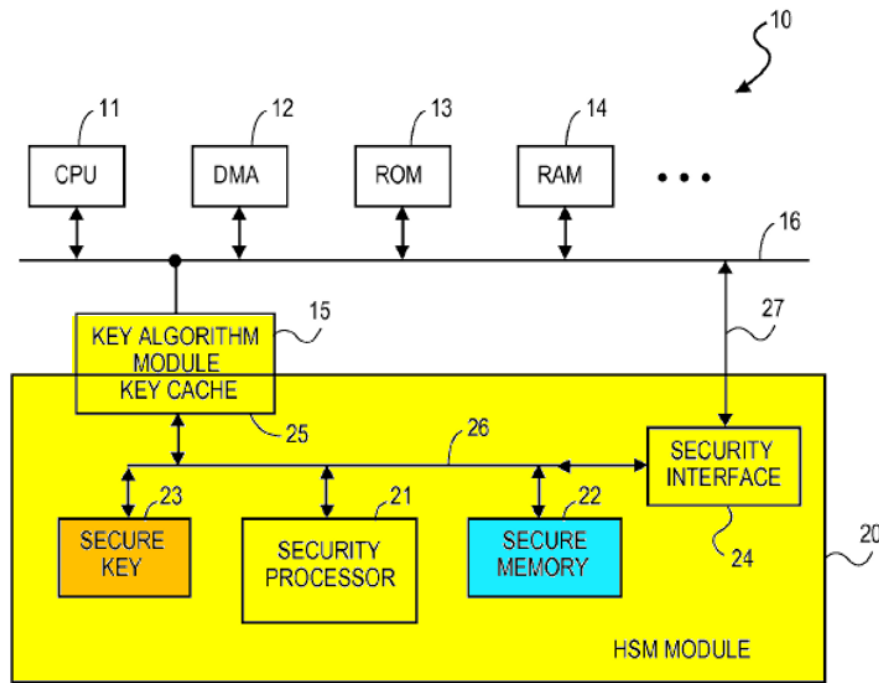


FIG. 1

Ex-1008, Fig. 1.

A POSITA would have understood that, in Buer's system modified to incorporate Chou's secure memory, Buer's encrypted session keys, which are one of the “one or more key value(s)...used by security processor **21** to encrypt and decrypt data,” can similarly be stored in secure memory **22**. Buer already provides

for non-volatile memory, used to store KEK and other private keys, and is accessible only to the security module. A POSITA would have found it obvious to expand the non-volatile memory to provide additional storage, such as Chou's secure memory, for encrypted keys that are used to encrypt and decrypt data. Ex-1002 ¶¶244-245. A POSITA would have been motivated to modify Buer's system to incorporate Chou's secure memory as explained in §XI.C.1.

h. 1[g]

Buer discloses establishing a secure session with a satellite security module over the network using a secure channel secured by the session key. Buer further explains that the satellite security module (the claimed "remote server") is located remotely from the main security module and "located throughout a data network (e.g., the Internet.>"). Ex-1009, 12:11-12; Ex-1002 ¶246.

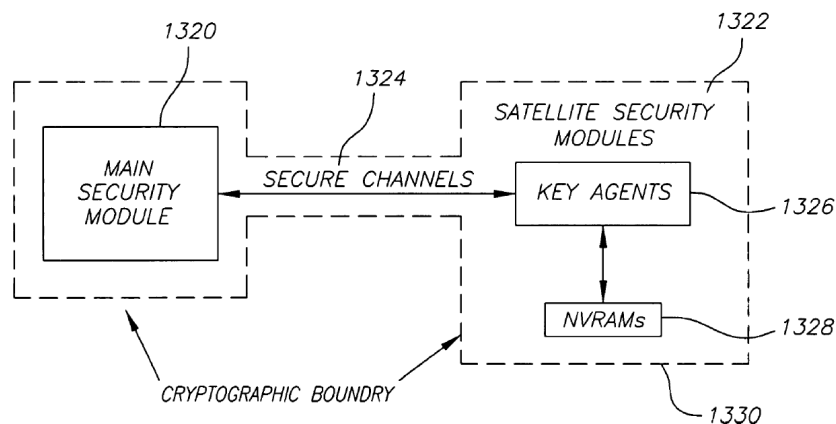


FIG. 13

Ex-1009, Fig. 13.

3. Dependent Claim 3

The '838 Patent claims encrypting a pre-master key with the security key, but does not describe this subject matter in its specification. Instead, it describes the normal TLS operation, i.e., encrypting a pre-master key with the server's public key. *See* Ex-1001, 8:28-32; 10:60-64; Ex-1002 ¶248. To the extent the specification's description is sufficient to disclose Claim 3, "encrypt the pre-master key *using the security key of the security engine*," Buer also discloses this element. Ex-1002 ¶249.

Buer discloses that the security module "uses the SSL protocol to provide data security in a network." Ex-1009, 6:13-15. Buer further explains that the security module receives the server's public key, and sends the "session information to the server encrypted with the server's public key." *Id.*, 6:26-29. "The keys to be used for the session are then derived from this information and the server's private key." *Id.*, 6:29-31. In other words, the "session information" is the shared information from which the session key is created, i.e., a "pre-master key." Thus, a POSITA would have understood that the "session information" encrypted with the server's public key and sent to the server according to the SSL protocol includes the pre-master secret from which the session keys are created. Ex-1002 ¶¶250-251.

4. Dependent Claim 4

(1) Disclosure in Buer (Ground 3)

As explained in Section **B.2.a** above, Buer discloses the main security module performing a cryptographic key exchange with a satellite security module (the remote server) SSL protocol, during which the main security module sends “session information” encrypted with server’s public key to the server. Also as explained above, a POSITA would have understood that the “session information” encrypted with server’s public key and sent to the server according to the SSL protocol includes a pre-master secret from which the session keys are created. Ex-1002 ¶252.

Buer also discloses that the wrapped pre-master key is stored in secure memory. For example, Buer discloses that a shared secret (such as the Diffie-Hellman shared secret generated during the cryptographic key exchange) can be stored in non-volatile memory associated with the security module. Ex-1009, 17:5-18 (“the non-volatile memory may comprise a one-time-programmable circuit for storing, for example, an initial value for KEK, a private key or a shared secret”); 4:58-60 (“the non-volatile memory **134** may be integrated into the key manager's integrated circuit to control access to the non-volatile memory **134**.”); Ex-1002 ¶253. The “shared secrets” stored in non-volatile memory associated with the security module are established through the cryptographic key exchange

process, i.e., they are the pre-master keys established through the key exchange.

Ex-1009, 13:14-20; Ex-1002 ¶254.

In addition, Buer discloses storing encrypted keys in a database as security associations, as explained in Section **XI.C.2.g.(1)** (limitation 1[f], “store the encrypted session key in the secure memory”). Ex-1002 ¶255. A POSITA would have understood that an encrypted pre-master key is also a “security association” to be stored in the security association database. *See* Ex-1009, 4:21-26 (“For convenience the term ‘security association’ will be used herein to refer to key information. This key information may include, for example, a key or keys, one or more encrypted keys and associated identifiers and other information such as rules relating to how to use the keys and the types of algorithms that may be used to decrypt the keys.”). Ex-1002 ¶255.

In addition, Buer acknowledges that “the encrypted security associations may be stored in any data memory.” This would include, for example, the non-volatile memory **134** connected to key manager 132 (see Fig. 1) or private keys store **330** attached to security module **320** (see Fig. 3). A POSITA would have recognized that, because Buer contemplates creating a session key from the pre-master key using the security module, it would be most efficient to store the session keys within the cryptographic boundaries of the security module or the cryptographic accelerator, thus eliminating the need for additional communication

with an otherwise unsecured host processor. Ex-1002 ¶¶256-257. Buer also explains that non-volatile memory can be made more secure by integrating into the same circuit as the key manager. Ex-1009, 17:1-4. Thus, A POSITA would have been motivated to store the encrypted pre-master keys inside the secure memory directly attached to the security module. Ex-1002 ¶¶256-259.

(2) Combination with Chou (Ground 4)

To the extent Buer does not disclose “stor[ing] the wrapped pre-master key in the secure memory,” combination with Chou renders this limitation obvious. Chou discloses an HSM, which includes a memory used to store data processed by the security processor. Ex-1008 ¶21. The wrapped pre-master key is one piece of data processed by the security module of Buer. Ex-1002 ¶¶260-261. Thus, in the proposed combination (Buer’s system, modified to include the secure memory of Chou to store data processed by the security module), the wrapped pre-master key would also be stored in the secure memory inside the security module. Ex-1002 ¶261. A POSITA would have recognized that Chou’s secure memory inside the HSM provides several benefits and would have been motivated to modify Buer’s system to incorporate Chou’s secure memory for the reasons explained above in Section XI.C.1. *Id.* ¶¶262, 205-212 (motivations to combine).

5. Dependent Claim 5

a. 5[pre]/5[a]

Buer discloses a cryptographic key exchange using the Diffie-Hellman algorithm. The “Diffie-Hellman shared secret” (K) created by both the Responder (in step 3 in the example below) and the Initiator (in step 7) is the pre-master key. The session key is generated as a function of the pre-master key according to the agreed-upon key generation method. Ex-1002 ¶¶263. For example, Buer describes the steps of the various exchanges, leading to the Initiator “deriv[ing] shared key material using the same formula” as the Responder. Ex-1009, 14:11-42. A POSITA would have understood that the shared secret is typically used as a pre-master secret from which a session key is derived according to a pre-agreed key derivation function (as the case would be in the SSL/TLS protocol). Ex-1002 ¶264.

b. 5[b]

Buer’s cryptographic key exchange algorithms include steps of calculating HMAC (Hash-based Message Authentication Code) values (i.e., calculate a hash function) to generate a key material stream message as a function of the nonce generated by the main security module (calculated at the beginning of the exchange) and a nonce received from the satellite security module. HMAC is a cryptographic hash function. Ex-1002 ¶265.

Buer's key exchange steps includes a Mutually Authenticated Diffie-Hellman exchange where "Responder" (i.e., the satellite security module) computes the "HMAC (KEK, xi || xr || Ni || Nr)." This means the HMAC of a message composed of four concatenated values (xi, xr, Ni, and Nr) is calculated using a secret key (KEK). Ex-1002 ¶266. The values Ni and Nr are nonces exchanged between the Initiator and the Responder. *Id.* ¶¶266-268. Once the Initiator receives the HMAC value from the responder, the Initiator "verifies the HMAC over the exchange parameters"—i.e., independently calculates the HMAC value using the parameters that it derived from the key exchange steps, and compare the result to the HMAC value received from the Responder. Ex-1002 ¶¶269; Ex-1009, 14:11-42 (step 7).

6. Dependent Claim 6

a. 6[pre]/6[a]

The "private Diffie-Hellman client key" is the random number xi, initially generated by the initiator (the client) at the beginning of the Diffie-Hellman algorithm (and kept private). Ex-1002 ¶¶ 35-50, 270. The "public Diffie-Hellman client key" is the value $y_i = g^{x_i} \bmod p$, which is generated using the private Diffie-Hellman client key (xi), the prime number (often referred to as "*p*"), and the primitive root modulo number (often referred to as "*g*"). *Id.* The value yi is sent to the responder via an unsecured channel, and is therefore "public." *Id.* ¶270.

Buer discloses generating a public Diffie-Hellman client key and a private Diffie-Hellman client key during the Diffie-Hellman key exchange between the main security module and the satellite security module. For example, Buer discloses “Initiator generates random number x_i , and creates public number $y_i = g^{x_i} \bmod p$ ” during the One-Way RSA Authenticated Diffie-Hellman key exchange. Ex-1009, 13:50-51; *see also*, 13:25-26; 14:19-20. Ex-1002 ¶¶271.

b. 6[b]

In the One-Way RSA Authenticated Diffie-Hellman exchange described in Buer, the “Initiator sends y_i [i.e., the public Diffie-Hellman client key] to the responder, along with a nonce N_i and its signature over the combination.” Ex-1009, 13:52-53 (emphasis added); Ex-1002 ¶¶271-272. This means the Initiator (the main security module) creates a digital signature by applying a cryptographic hash function to the combination of the public Diffie-Hellman client key (y_i) and the nonce (N_i), using the Initiator’s private key. Ex-1002 ¶¶71-76 (explanation of digital signature). As quoted above, Buer further explains that the digital signature is “signed by the root key to establish the trusted public key at the satellite security module” and the “root key” is “loaded at manufacturing time” i.e., the claimed “security key.” Ex-1009, 13:66-14:6.

c. 6[c]

As explained above, in step 2 of the One-Way RSA Authenticated Diffie-Hellman described in Buer, the “Initiator sends y_i [i.e., the public Diffie-Hellman client key] to the responder, along with a nonce N_i and its signature over the combination.” Ex-1009, 13:52-53 (emphasis added); Ex-1002 ¶273.

7. Dependent Claim 7

a. 7[pre]/7[a]

During a Diffie-Hellman key exchange, the server (the Responder in a Diffie-Hellman exchange) also generates a random number x_r , which is kept private, and calculates $y_r = g^{x_r} \bmod p$. This value y_r , which is then shared with the client (the Initiator), is referred to as the “public Diffie-Hellman server key.” Ex-1002 ¶¶274, 35-50.

Buer discloses Responder (the satellite security module) generating a public Diffie-Hellman server key during the Diffie-Hellman key exchange between the main security module and the satellite security module, and sending it to the Initiator (the main security module, i.e., the claimed “remote server”). In the One-Way RSA Authenticated Diffie-Hellman example in Buer, the “Responder verifies the signature and generates a random number x_r , public number $y_r = g^{x_r} \bmod p$,” (step 3) and “returns y_r to the initiator, along with nonce N_r ” (step 6). Ex-1009, 13:45-65. Thus, the Initiator receives the public Diffie-Hellman server key (y_r) from the Responder (the remote server). Ex-1002 ¶¶274-275.

b. 7[b]

Buer further explains that the Responder generates “the Diffie-Hellman shared secret, $K=y_i^{x_r} \bmod p$ ” (step 3), and that “Initiator derives shared key material using the same formula” (step 7). Ex-1009, 13:45-65. A POSITA would understand that the shared key material ($K=y_i^{x_r} \bmod p$) generated using the same formula on the Initiator side ($K=y_r^{x_i} \bmod p$) will yield the same result (thereby establishing the pre-master secret K). In the formula used by the Initiator, y_r is the public Diffie-Hellman server key, and x_i is the private Diffie-Hellman client key. Ex-1002 ¶276.

If the shared secret K is directly used as the session key, it is evident from the formula for K that the session key is generated as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key. Ex-1002 ¶277.

A POSITA would have also understood that, to the extent the shared secret K is not directly used as the session key, but used as a pre-master secret from which a session key is derived according to a pre-agreed key derivation function (as the case would be in the SSL/TLS protocol), the session key is generated using the pre-master key K , which is in turn generated using the private Diffie-Hellman client key and the public Diffie-Hellman server key. Thus, the session key is

generated as a function of the public Diffie-Hellman server key and the private Diffie-Hellman client key. Ex-1002 ¶278.

8. Claim 24

Buer's security module and/or cryptographic accelerator is communicatively coupled to the host processor through an interconnect.

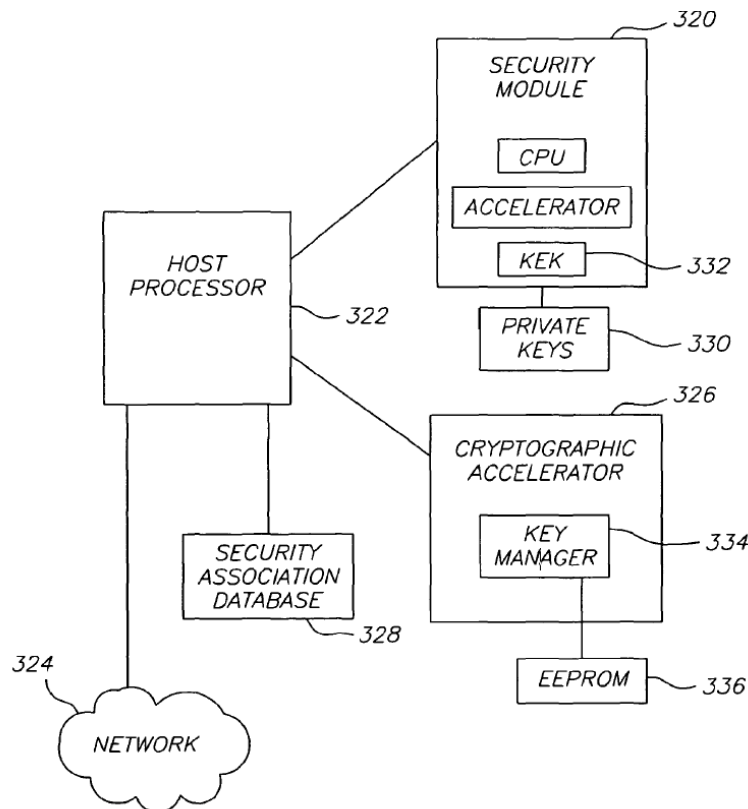


FIG. 3

Ex-1009, Fig. 3; *see also*, Figs. 1, 2, 4, 5, 10, 13. Buer also explains that “the connections represented by the lead lines in the drawings...may be made with physical wire, fiber and/or wireless connections.” Ex-1009, 16:47-62. Buer further explains that the various components, such as the security module and the

host processor, may be “combined on one or more integrated circuits,” i.e., forming an SOC. Ex-1009, 16:45-46; Ex-1002 ¶279.

9. Independent Claims 9, 17

a. 9[pre], 17[pre]

Buer states it discloses a method. Ex-1009, Abstract (“Methods and systems are disclosed for providing secured data transmission and for managing cryptographic keys.”); Ex-1002 ¶280.

Buer also discloses the preamble of Claim 17. For example, Buer states its invention relates “to systems and methods for providing secured data transmission and for managing cryptographic keys.” Ex-1009, 1:6-9. “[A] variety of hardware and software processing components may be used to implement the functions of the main security module[]” or the host processors. *Id.*, 16:36-46. A POSITA would have understood that such systems using software components would necessarily include one or more non-transitory machine-readable storage media, which stores the instructions that perform the disclosed methods when executing the disclosed system of Buer’s invention. Ex-1002 ¶281.

b. 9[a], 17[a]

See §XI.C.2.b, 1[a] (“...a security engine that is separate from a processor core of the system-on-a-chip...”) and §XI.C.2.c, 1[b] (the security engine to “generate a random nonce...”); Ex-1002 ¶¶282, 214 (1[a]), 217-222 (1[b]).

c. 9[b], 17[b]

See §**XI.C.2.c**, 1[b]; Ex-1002 ¶¶283, 217-222 (1[b]).

d. 9[c], 17[c]

See §**XI.C.2.d**, 1[c]; Ex-1002 ¶¶284, 223 (1[c]).

e. 9[d], 17[d]

See §**XI.C.2.e**, 1[d]; Ex-1002 ¶¶285, 224-229 (1[d]).

f. 9[e], 17[e]

See §**XI.C.2.f**, 1[e] (“encrypt the symmetric session key based on the security key”); §**XI.C.2.b**, 1[a] (“...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”); Ex-1002 ¶¶286, 230-233 (1[e]), 216 (1[a]).

g. 9[f], 17[f]

See §**XI.C.2.g**, limitation 1[f]; Ex-1002 ¶¶287, 234-245 (1[f]).

h. 9[g], 17[g]

See §**XI.C.2.h**, 1[g]; Ex-1002 ¶¶288, 246 (1[g]).

10. Dependent Claims 10, 18

a. 10[pre], 18[pre].

Buer discloses that the security module “uses the SSL protocol to provide data security in a network.” Ex-1009, 6:13-15. The SSL protocol includes standard handshake messages, which begin with a ClientHello message that includes a nonce generated by the client. Ex-1002 ¶¶61-70, 289. In the context of

Buer, the initial message during the key exchange protocol steps sent by the main security module (called the “Initiator”) to the satellite security module (called the “Responder”) constitutes the ClientHello message. Ex-1002 ¶290. In the Unauthenticated Diffie-Hellman exchange example in Buer, the initiator (i.e., the main security module) sends the first message (a ClientHello message) that includes the nonce generated by the initiator to the responder (the claimed “remote server”). Ex-1009, 13:21-44 (“Initiator sends y_i to the responder, along with a nonce N_i ” at step 2). Ex-1002 ¶291. Buer discloses similar steps of sending the initial message including nonce N_i in each of the other exemplary key exchange steps. *See* Ex-1009, 13:45-14:10 (One-Way RSA Authenticated Diffie-Hellman beginning with a message that includes nonce N_i); 14:11-42 (Mutually Authenticated Diffie-Hellman); 14:43-59 (Shared Secret ReKey). Ex-1002 ¶292.

b. 10[a], 18[a]

In response to the initial message from the main security module (the “Initiator”) (the ClientHello message), the satellite security module (the “Responder”) sends a response message that includes a Responder-generated nonce N_r . This message constitutes the ServerHello message. Ex-1002 ¶293. In the Unauthenticated Diffie-Hellman exchange, the Initiator receives the response from the Responder (a ServerHello message) that includes the nonce generated by the Responder. Ex-1009, 13:21-44 (“Responder generates...nonce N_r ” (step 4)

and “returns yr to the initiator, along with nonce Nr” (step 6)); Ex-1002 ¶294.

Buer discloses similar steps of the Responder sending a message including nonce Nr in each of the other exemplary key exchange steps. *See* Ex-1009, 13:45-14:10 (One-Way RSA Authenticated Diffie-Hellman); 14:11-42 (Mutually Authenticated Diffie-Hellman); 14:43-59 (Shared Secret ReKey); Ex-1002 ¶295.

c. 10[b], 18[b]

(1) Disclosure in Buer (Ground 3)

Buer discloses data memory locations that can be used by the security engine. These data locations include, for example, non-volatile memory **134** connected to key manager **132** (see Fig. 1) or private keys store **330** attached to security module **320** (see Fig. 3). Ex-1002 ¶296. A POSITA would have recognized that, because Buer contemplates generating session keys from the data received through the cryptographic key exchange using the security module, it would be most efficient to store the data, such as the server nonce, within the cryptographic boundaries of the security module, thus eliminating the need for additional communication with an otherwise unsecured host processor. *Id.*

A POSITA would have further recognized that, storing the server nonce, which is further used by the security module to generate the shared key material, would eliminate the need to manage such data in a secure manner outside the security module. Ex-1002 ¶297. Because Buer’s non-volatile memory is already

access controlled by being coupled to the security module, separate management or access control to the security association database is not needed. *Id.* Thus, from Buer's disclosure of non-volatile memory coupled to the security module, a POSITA would have been motivated to store the data used by the security module, such as the server nonce in such memory. *Id.*

(2) Combination with Chou (Ground 4)

Chou discloses storing the data to be processed by the security processor in secure memory. For example, Chou discloses a Hardware Security Module (HSM), which includes a memory used to store data processed by the security processor. Ex-1008 ¶21. In Buer's system modified to incorporate Chou's secure memory into the security module, server nonce (the nonce Nr received from the satellite security module) is one piece of data processed by the security module. Thus, in view of Chou's disclosure that data processed by the security module is to be stored in secure memory, Buer's system modified to incorporate Chou's secure memory would store the server nonce in the secure memory inside the security module. Ex-1002 ¶¶298-299.

A POSITA would have recognized that Chou's secure memory inside HSM provides several benefits and would have been motivated to modify Buer's system to incorporate Chou's secure memory for the reasons explained in Section **XI.C.1**. Ex-1002 ¶300, 205-212 (motivation to combine).

11. Dependent Claims 11, 19

See §XI.C.3, Cl. 3; Ex-1002 ¶¶301-303, 248-251 (Cl. 3).

12. Dependent Claims 12, 20

See §XI.C.4, Cl. 4; Ex-1002 ¶¶304-305, 252-262 (Cl. 4).

13. Dependent Claims 13, 21

See §XI.C.5.a, 5[pre]/5[a]; Ex-1002 ¶¶306, 263-264 (5[pre]/5[a]).

14. Dependent Claims 14, 22

See §XI.C.5.b, 5[b]; Ex-1002 ¶¶307. 265-269 (5[b]).

15. Claims 15, 23

a. 15[pre]/15[a], 23[pre]/23[a]

See §XI.C.6.a, 6[pre]/6[a]; Ex-1002 ¶¶308, 270-271 (6[pre]/6[a]).

b. 15[b], 23[b]

See §XI.C.6.b, 6[b]; Ex-1002 ¶¶309, 272 (6[b]).

c. 15[c], 23[c]

See §XI.C.6.c, 6[c]; Ex-1002 ¶¶310, 273 (6[c]).

d. 15[d], 23[d]

See §XI.C.7.a, 7[pre]/7[a]; Ex-1002 ¶¶311, 274-275 (7[pre]/7[a]).

e. 15[e], 23[e]

See §XI.C.7.b, 7[b]; Ex-1002 ¶¶312, 276-278 (7[b]).

D. Grounds 5 and 6: Grounds 3 and 4, in further combination with TLS-1.0

To the extent Buer, alone (Ground 3) or in combination with Chou (Ground 4) do not disclose all limitations of the challenged claims, those limitations are obvious in further combination with TLS-1.0. The combination would retain the structure of Buer, alone or in combination with Chou, while implementing the key exchange steps between the main and the satellite security modules using TLS-1.0.

1. Motivation to Combine

Buer notes that its “security module [] uses the SSL protocol to provide data security in the network.” Ex-1009, 6:13-15; Ex-1002 ¶314. TLS-1.0, the successor to the SSL protocol, specifies a substantially similar protocol to SSL v.3.0, and commonly referred to together as SSL/TLS protocol. Ex-1002 ¶¶61-70. Thus, to the extent not already disclosed in Buer itself by reference to the SSL protocol, a POSITA would have been motivated to incorporate the specific teachings of TLS-1.0 in the communication between the main security module and the satellite security module. Ex-1002 ¶314; *see also, Intel v. PACT*, 2023 WL 2469631, at *5.

A POSITA would have recognized the various key exchange steps disclosed in Buer can be implemented using the standard TLS handshake procedure, and would have been motivated to incorporate the TLS handshake protocol into Buer’s cryptographic key exchange steps. Ex-1002 ¶315. As explained in Section **XI.B.1**

above, a POSITA would have been motivated to use TLS-1.0 because doing so would allow the implementer to use any one of the many available SSL/TLS open source libraries, to save development resources, to ensure interoperability with other systems, and to take advantage of test and development efforts by open source developers. *Id.* ¶316. A POSITA could have reasonably expect success in implementing TLS-1.0 into Buer's system based on the large support network of open source implementers, extensive testing and revisions based on peer review and resulting quality and reliability, and development experience of open source communities. *Id.* ¶317.

2. Dependent Claim 3

TLS-1.0 specifies the use of an RSA-encrypted premaster secret message that discloses the client device (such as the main security module of Buer) creating a premaster secret message, which requires generating a pre-master key, encrypting the pre-master key using the public key of the server (received in a server certificate), and sending the encrypted pre-master key to the server. Ex-1002 ¶¶318-319.

7.4.7.1. RSA encrypted premaster secret message

Meaning of this message:

If RSA is being used for key agreement and authentication, the client generates a 48-byte premaster secret, encrypts it using the public key from the server's certificate or the temporary RSA key provided in a server key exchange message, and sends the result in an encrypted premaster secret message. This structure is a variant of the client key exchange message, not a message in itself.

Ex-1006, §7.4.7.1.

3. Dependent Claim 4

To the extent the security engine “to wrap the pre-master key using a server public key” is not explicitly disclosed in Buer, combination with TLS-1.0 discloses it. *See* §**XI.D.2** (Cl. 3 above); §**XI.D.1**; Ex-1002 ¶¶320-322. Buer and Chou disclose all remaining elements of Claim 4. *See* §**XI.C.4** (Grounds 3/4).

4. Dependent Claim 5

To the extent the “generate the session key as a function of the pre-master key” is not explicitly disclosed in Buer, combination with TLS-1.0 discloses it. *See* §**XI.B.4.a** (Ground 2, disclosure of 5[a] in TLS-1.0); Ex-1002 ¶¶323-328. Buer and Chou disclose all remaining elements of Claim 5. *See* §**XI.C.5** (Grounds 3/4).

5. Dependent Claim 7

To the extent “generate the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key” is not explicitly disclosed in Buer, combination with TLS-1.0 discloses it. *See* §**XI.B.6.b** (Ground

2, disclosure of Claim 7[b] in TLS-1.0); Ex-1002 ¶¶329-334. Buer and Chou disclose all remaining elements of Claim 7. *See* §**XI.C.5** (Grounds 3/4).

6. Dependent Claims 10, 18

a. 10[pre], 18[pre]

To the extent Buer does not explicitly use the term “ClientHello” to refer to the initial message from the Initiator to the Responder, a POSITA would have found this limitation obvious in light of TLS-1.0. *See* §**XI.B.7.a** (Ground 2, disclosure of 10[pre]/18[pre] in TLS-1.0); Ex-1002 ¶¶335-337.

b. 10[a], 18[a]

To the extent Buer does not explicitly use the term “ServerHello” to refer to the Responder’s message sent in response to the ClientHello message, a POSITA would have found this limitation obvious in light of TLS-1.0. *See* §**XI.B.7.b** (Ground 2, disclosure of 10[a]/18[a] in TLS-1.0); Ex-1002 ¶¶338-341.

c. 10[b], 18[b]

See §**XI.C.10.c**, disclosure in Buer and/or Chou; Ex-1002 ¶¶296-300 (Grounds 3/4).

7. Dependent Claims 11, 19

See §**XI.D.2**, Cl. 3; Ex-1002 ¶¶301-303, 318-319 (Cl. 3).

8. Dependent Claims 12, 20

To the extent “wrapping the pre-master key...using a server public key” (12[a] and 20[a]) is not explicitly disclosed in Buer, TLS-1.0 discloses it. *See*

§**XI.D.3**, Cl. 4 (“...wherein the security engine is further to wrap the pre-master key using a server public key....”); §**XI.D.1**; Ex-1002 ¶¶344-345, 320-322 (Cl. 4). Buer alone or in combination with Chou discloses all other limitations of Claims 12 and 20. *See* §**XI.C.12**.

9. Dependent Claims 13, 21

See §**XI.D.4**, 5[a]; Ex-1002 ¶¶323-328 (5[a]).

10. Dependent Claims 14, 22

See §**XI.C.5.b**, (Ground 3/4, 5[b]), Ex-1002 ¶¶347, 265-269 (Ground 3/4, 5[b]).

11. Claims 15, 23

To the extent Buer does not disclose generating the session key as a function of the private Diffie-Hellman client key and the public Diffie-Hellman server key (15[e], 23[e]), TLS-1.0 discloses it. *See* §**XI.D.5**, 7[b]; §**XI.D.1**; Ex-1002 ¶348, 329-334 (Cl. 7). Buer alone or in combination with Chou discloses all other limitations of Claims 15 and 23. *See* §**XI.C.15**; Ex-1002 ¶349.

XII. THE BOARD SHOULD NOT USE ITS DISCRETION TO DENY INSTITUTION UNDER *FINTIV*

The Board should decline to exercise its discretion to deny institution under 35 U.S.C. § 314(a). While Director Vidal’s June 21, 2022 Guidance made clear that “*Fintiv* is limited to facts of that case,” for completeness, Petitioner addresses the factors set forth in *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (PTAB

Mar. 20, 2020) (precedential). As explained below, the six factors set out in *Fintiv* do not justify denying institution.

The **first *Fintiv* factor** is neutral because Samsung has not yet moved to stay the district court proceedings pending IPR, and the PTAB will not infer how the district court would rule should a stay be requested. *See, e.g., Hulu, LLC v. SITO Mobile R&D IP, LLC*, IPR2021-00298, Paper 11 at 10–11 (PTAB May 19, 2021).

For the **second *Fintiv* factor**, the district court has set a trial date in the Daedalus matter for May 20, 2024. This timing does not tip in favor of denying the petition in this case. *See, e.g., Apple Inc. v. Koss Corp.*, IPR2021-00305, Paper 14 at 13 (PTAB June 3, 2021). This Petition is filed in March 2023. An institution decision should be issued within six months of docketing, around September 2023. And a final written decision (FWD) should issue in September 2024. Thus, the trial date is likely about four months before the Final Written Decision (“FWD”). Recent PTAB cases analyzing similar schedules have weighed the second factor as neutral or, at most, slightly in favor of denial. *See, e.g., Google LLC v. Parus Holdings, Inc.*, IPR2020-00847, Paper 9 at 12 (PTAB Oct. 21, 2020) (trial scheduled for three months before FWD weighs “at best, neutral”); *Apple Inc. v. Koss Corp.*, IPR2021-00305, Paper 14 at 13-14 (PTAB June 3, 2021) (trial scheduled for two months before FWD weighs “at most, minimally in favor” of denial).

“The PTAB will also consider additional supporting factors such as the number of cases before the judge in the parallel litigation.” Director Vidal’s June 2022 Guidance. Here, the court has scheduled three other trials (2:22-cv-00312-JRG-RSP, *Fleet Connect Solutions LLC v. Southern Tire Mart, LLC*; 2:22-cv-00322-JRG-RSP, *Lionra Technologies Limited v. Fortinet, Inc.*; and 2:22-cv-00361-JRG-RSP S3G, *Technology LLC v. Nordstrom Inc.*) in the same week as the trial for this case.

If the median time to trial for civil actions in the Eastern District of Texas is used to estimate the trial date, as advised by Director Vidal’s June 2022 Guidance, trial should not be expected to occur until October 2024, a month *after* a FWD. Ex-1024, 35 (median time to trial for the EDTX is 24.5 months).

The **third *Fintiv* factor** weighs strongly against denial. In particular, Samsung has not yet answered Daedalus’s Complaint, and has instead moved to dismiss. No hearing has been set and no substantive activity has occurred in the case. Because the investment in the trial has been minimal and Petitioner acted diligently, this factor weighs strongly against discretionary denial. *See, e.g., Hulu*, Paper 11 at 13.

The **fourth *Fintiv* factor** also weighs against denial. As explained above, the median time to trial for civil actions in EDTX suggests that trial will not occur until at least a month *after* a FWD. Thus, “there is a reasonable likelihood that the

Board will address the overlapping validity issues prior to the district court reaching them at trial..., thereby providing the possibility of simplifying issues for trial.” *Juniper Networks, Inc. v. Packet Intelligence LLC*, IPR2020-0039, Paper 21 at 18 (PTAB Sept. 10, 2020); *see also MED-EL Elektromedizinische Geräte GmbH v. Sonova AG*, IPR2020-00176, Paper 13 at 15 (PTAB June 3, 2020) (“As to the fourth factor, the parties do not dispute that overlap exists between the invalidity issues in this case and in the district court. This overlap may inure to the district court’s benefit, however, by simplifying issues for trial should we reach our determination on the challenges raised in the Petition before trial.”).

On the **fifth *Fintiv* factor**, the Board should give little weight to the fact that Petitioner and PO are the same parties as in district court. Although Samsung is both petitioner and litigation defendant, this factor does not outweigh the other factors that strongly weigh against discretionary denial. *See Weatherford U.S., L.P., v. Enventure Global Tech., Inc.*, IPR2020-01684, Paper 16 at 11–13 (PTAB Apr. 14, 2021).

“Other circumstances” under the **sixth *Fintiv* factor** further weigh against discretionary denial. The Petition’s merits are strong with two primary references, Chou and Buer, alone or in combination with one secondary reference, rendering independent Claims 1 and 14 and dependent Claims 2, 3, and 11 obvious. *See Align Technology, Inc. v. 3Shape A/S*, IPR2020-01087, Paper 15 at 42–43 (PTAB

Jan. 20, 2021); *see also supra* Section VII.

Taken together holistically, the *Fintiv* factors strongly weigh against exercising discretion to deny institution.

XIII. CONCLUSION

For the reasons given above, Petitioner requests institution of IPR for Claims 1, 3-7, 9-15, 17-24 of the '838 Patent based on each of the grounds specified in this Petition.

Respectfully Submitted,

/s/ John Kappos
John Kappos (Reg. No. 37,861)

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), Petitioner certifies that this petition includes 13,774 words, as measured by Microsoft Word, exclusive of the table of contents, mandatory notices under § 42.8, certificates of service, word count, and exhibits.

CERTIFICATE OF SERVICE (37 C.F.R. § 42.6(e)(1))

The undersigned hereby certifies that the above document was served on March 20, 2023, by filing this document through the Patent Trial and Appeal Board P-TACTS System, as well as delivering a copy via express mail upon the following attorneys of record for the Patent Owner and Petitioner:

Barnes & Thornburg LLP
11 S. Meridian Street
Indianapolis, IN 46204

A courtesy copy was sent to the below counsel via electronic mail:

Garland T. Stephens
Blue Peak Law Group
2007 South Boulevard
Houston, TX 77098
Phone: (832) 387-7696
Email: garland@bluepeak.law

Anna Dwyer
Blue Peak Law Group
345 East 94th Street, Ste Apt 28g
New York, NY 10128
Phone: (772) 631-6460
Email: anna@bluepeak.law

Jeff Risher
Blue Peak Law Group
15226 Bluff View Ave.
Friant, CA 93626
Phone: (949) 690-1462
Email: jeff@bluepeak.law

Justin Lile Constant
Blue Peak Law Group
3139 W Holcombe Blvd., PMB 8160

Houston, TX 77025
Phone: (281) 972-3036
Email: justin@bluepeak.law

Kate S. Falkenstien
Robert Stephen Magee
Blue Peak Law Group
3790 El Camino Real, Ste PMB 846
Palo Alto, CA 94306
Phone: (281) 972-3036
Email: kate@bluepeak.law
Email: robert@bluepeak.law

Richard M. Koehl
Blue Peak Law Group
PO Box 20204
New York, NY 10001-0006
Phone: (917) 856-3872
Email: richard@bluepeak.law

Andrea Leigh Fair
Claire Abernathy Henry
Charles Everingham , IV
Ward, Smith & Hill, PLLC
1507 Bill Owens Parkway
Longview, TX 75604
Phone: (903) 757-2323
Email: andrea@wsfirm.com
Email: claire@wsfirm.com
Email: ce@wsfirm.com

Michael T. Renaud
Mintz Levin Cohn Ferris Glovsky & Popeo PC
One Financial Center, Suite 100
Boston, MA 02111
Phone: (617) 542-6000
Fax: (617) 542-2241
Email: mtrenaud@mintz.com

Dated: March 20, 2023

Respectfully submitted,

/s/ John Kappos

John Kappos (Reg. No. 37,861)
O'Melveny & Myers LLP
2501 North Harwood Street, Suite 1700
Dallas, TX 75201
Telephone: (972) 360-1900
Fax: (972) 360-1901
E-Mail: jkappos@omm.com
Attorney for Petitioner