

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NXP USA, INC.,
Petitioner,

v.

MEDIATEK INC.,
Patent Owner.

Case IPR2022-00759
Patent 8,464,037 B2

PETITION FOR *INTER PARTES* REVIEW UNDER 35 U.S.C. § 312

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	MANDATORY NOTICES	2
	A. Real Party in Interest (37 C.F.R. § 42.8(b)(1))	2
	B. Related Matters (37 C.F.R. § 42.8(b)(2)).....	2
	C. Lead and Back-Up Counsel (37 C.F.R. § 42.8(b)(3)).....	2
	D. Service Information (37 C.F.R. § 42.8(b)(4))	3
III.	GROUND FOR STANDING.....	3
IV.	STATEMENT OF PRECISE RELIEF REQUESTED FOR EACH CLAIM CHALLENGED	4
	A. Claims for Which Review Is Requested: 1-3, 5-6, 19-21	4
	B. Statutory Grounds of Challenge (37 C.F.R. § 42.104(b)(2))	4
V.	FIELD OF TECHNOLOGY	5
	A. Secure Hardware	6
	B. Secure Boot	7
VI.	LEVEL OF ORDINARY SKILL IN THE ART.....	7
VII.	THE '037 PATENT.....	8
	A. Background and Perceived Shortcomings	8
	B. The '037 Patent's Asserted Improvement.....	10
	C. Prosecution History	13
	D. Claim Construction (37 C.F.R. § 42.104(b)(3)).....	13
VIII.	SPECIFIC EXPLANATION OF GROUNDS	14
	A. Ground 1 – Claims 19-21, 1-3, and 4-5 Are Obvious Over Smith.....	14
	1. Overview of Smith.....	14
	2. Element-by-Element Analysis	15
	a) Limitations 19[pre] and 19[a].....	15
	(1) EDRAM 111 of chip 105 is a volatile random access memory	16

(2)	ROM 113 of chip 105 is a non-volatile memory.....	17
(3)	Chip 105 obviously would have included a CPU core	17
(4)	Chip 105 obviously would have included a bus system	19
(5)	“Chip” 105 included volatile memory and non-volatile memory as well as, obviously, a processor core and a bus system, all of which would have therefore been on a substrate.....	21
b)	Limitation 19[b].....	21
(1)	Certain codes 115	21
(2)	UID 119	24
c)	Claim 20.....	30
d)	Limitation 21[pre].....	32
e)	Limitation 21[a].....	33
f)	Limitations 21[b][i]-21[b][ii]	34
g)	Limitation 21[c].....	34
h)	Limitation 1[pre].....	34
i)	Limitation 1[a]	35
j)	Limitation 1[b].....	39
k)	Limitation 1[c].....	39
l)	Limitation 1[d].....	40
m)	Limitation 1[e]	41
n)	Limitation 1[f]	41
o)	Claim 2.....	42
p)	Claim 3.....	42
q)	Claim 5.....	47
r)	Claim 6.....	47

B.	Ground 2 – Claim 20 Is Obvious Over Smith in View of Bolotin	51
1.	Overview of Bolotin	51
2.	Motivation to modify Smith’s chip 105 chip to include circuitry like that provided by Bolotin’s micro-controller 212 and locking function controller 215	53
3.	Smith’s chip 105, modified in view of Bolotin, meets claim 20	54
C.	Grounds 3 and 4 – Claims 19 and 21 Are Obvious Over Smith in View of Little (Ground 3), and Claim 20 Is Obvious Over Smith in View of Little and Bolotin (Ground 4).....	55
D.	Ground 5 – Claims 5-6 Are Obvious Over Smith in view of Challenger	56
1.	Overview of Challenger.....	56
2.	Motivation to Combine	57
3.	Modification of Smith in view of Challenger	58
4.	Element-by-Element Analysis	61
a)	Claim 5.....	61
b)	Claim 6.....	61
IX.	DISCRETIONARY DENIAL UNDER SECTION 325(d) IS IMPROPER	62
A.	Prosecution History	62
B.	The Grounds Are Not Based on Substantially the Same Prior Art or Arguments Previously Considered by the Office.....	63
X.	CONCLUSION.....	64

PETITIONER'S EXHIBIT LIST

Exhibit	Shorthand	Description
1001	'037 Patent	U.S. Patent No. 8,464,037 B2
1002	Madisetti	Declaration of Dr. Vijay Madisetti
1003	Challener	U.S. Patent App. Pub. No. US 2003/0041254 A1
1004	Smith	U.S. Patent App. Pub. No. US 2008/0165952 A1
1005	Stevens	U.S. Patent No. 6,663,976 B1
1006	Bolotin	WIPO Int'l Pub. No. WO 2007/014074 A1
1007	Petković	Petković and Jonker (Eds.), <i>Security, Privacy, and Trust in Modern Data Management</i> , Springer (2007)
1008	Špánek	Špánek, <i>Self-organizing and Self-monitoring Security Model for Dynamic Distributed Environments</i> (February 2008)
1009	OECD	OECD (2008-01-14), <i>Radio-Frequency Identification (RFID): A Focus on Information Security and Privacy</i> , OECD Digital Economy Papers, No. 138, OECD Publishing, Paris.
1010	Amazon	November 1, 2007 archived copy of https://www.amazon.com/Security-Privacy-Management-Data-Centric-Applications/dp/3540698604 from https://web.archive.org/web/20071101011130/https://www.amazon.com/Security-Privacy-Management-Data-Centric-Applications/dp/3540698604
1011	Archive	Declaration of Nathaniel E Frank-White of the Internet Archive
1012	Little	U.S. Patent App. Pub. No. US 2006/0112266 A1
1013	Chen	U.S. Patent No. 7,546,447 B2
1014	Bhatt	U.S. Patent App. Pub. No. US 2006/0107321 A1
1015	Kumar	U.S. Patent App. Pub. No. US 2008/0159541 A1
1016	'037 PH	Prosecution history of U.S. Patent No. 8,464,037 B2

Exhibit	Shorthand	Description
1017	Microsoft	Excerpts of <i>Microsoft Computer Dictionary</i> , Fifth Edition (2002)
1018	Dyer	U.S. Patent App. Pub. No. US 2009/0236691 A1
1019	Fadell	U.S. Patent App. Pub. No. US 2004/0224638 A1

Claim Listing

Claim	Limitation No.	Limitation
1	1[pre]	A method for booting a computer system, the method comprising:
	1[a]	upon one of a power-on and a reset event, executing instructions stored in an internal non-volatile memory area of a central processing unit,
	1[b]	said internal non-volatile memory area containing an encryption key;
	1[c]	determining a current signature of at least a signed portion of boot data,
	1[d]	said signed portion of boot data containing a stored signature of said portion of boot data,
	1[e]	said stored signature created by applying said encryption key; and
	1[f]	executing boot instructions contained in said signed portion of boot data, when said current signature and said stored signature are identical.
2	2[pre]	The method of claim 1, wherein executing said instructions stored in said internal non-volatile memory area comprises
	2[a]	initializing an internal random access memory of said central processing unit and loading said at least a portion of boot data from a non-volatile memory into said initialized random access memory.
3	3[pre]	The method of claim 2, wherein executing boot instructions of said signed portion of boot data comprises
	3[a]	initializing a system memory of said computer system and loading said boot data from said non-volatile memory into said system memory and verifying an integrity of said boot data.
5	5	The method of claim 1, further comprising locking said encryption key for external access prior to executing said boot instructions.
6	6	The method of claim 5, further comprising determining whether an upgrade version of said boot data is available.

Claim	Limitation No.	Limitation
19	19[pre]	A central processing unit (CPU), comprising:
	19[a]	a substrate having formed thereon circuit elements defining a CPU core, a volatile random access memory, a non-volatile memory and a bus system for connecting said CPU core, said volatile random access memory and said non-volatile memory; and
	19[b]	pre-boot information stored in said non-volatile memory, said pre-boot information including instructions executable by said CPU core and a symmetric encryption key for verifying at least a portion of a boot routine signed by using said symmetric encryption key.
20	20	The central processing unit of claim 19, further comprising a locking mechanism configured to enable a portion of said non-volatile memory containing said symmetric encryption key upon one of a reset event and a power-on event only.
21	21[pre]	A computer system, comprising:
	21[a]	a central processing unit comprising:
	21[b][i]	a substrate having formed thereon circuit elements defining a CPU core, a volatile random access memory, a non-volatile memory and a bus system for connecting said CPU core, said volatile random access memory and said non-volatile memory, and
	21[b][ii]	pre-boot information stored in said non-volatile memory, said pre-boot information including instructions executable by said CPU core and a symmetric encryption key for verifying at least a portion of a boot routine signed by using said symmetric encryption key; and
	21[c]	a non-volatile boot memory configured to hold at least said boot routine and a second boot routine.

I. INTRODUCTION

The '037 Patent discloses using an encryption key stored in a non-volatile memory of a central processing unit (“CPU”) to implement a secure boot mechanism in a computer system. '037 Patent, Abstract. The secure boot mechanism uses the encryption key upon either a reset or power-on event to determine a signature of boot data, and then compares the determined signature to a stored signature created using the same encryption key in order to verify the integrity of the boot data. '037 Patent, Abstract, 3:38-42. Each of the challenged claims requires an encryption key that is used to verify at least a portion of a boot routine signed using the encryption key. The challenged claims, however, are rendered obvious in view of Smith, which discloses or suggests to a POSITA a computer system, including volatile and non-volatile memories communicating via a bus system with a processor core on a central processing unit of the system, for deriving a signature for a code image using a symmetric encryption key, and then comparing the derived signature to a stored signature to verify the code image. As described below, Smith's operations involving a symmetric encryption key support verification of boot data in the same manner as the challenged claims.

II. MANDATORY NOTICES

A. Real Party in Interest (37 C.F.R. § 42.8(b)(1))

The real parties-in-interest are Petitioner, NXP Semiconductors N.V., Avnet, Inc., Arrow Electronics, Inc., Robert Bosch GmbH, Robert Bosch LLC, Continental AG, and Continental Automotive GmbH.

B. Related Matters (37 C.F.R. § 42.8(b)(2))

MediaTek Inc. v. NXP Semiconductors N.V., Civil Action No. 2:21-cv-4969-GW-AFM (C.D. Cal. 2021) (the “Litigation”) involves the ’037 Patent, U.S. Patent No. 9,265,056, U.S. Patent No. 10,211,948, U.S. Patent No. 6,738,845, U.S. Patent No. 6,823,451, and U.S. Patent No. 9,538,531 asserted by MediaTek Inc., and U.S. Patent No. 9,480,018, U.S. Patent No. 10,278,224, U.S. Patent No. 10,404,839, and U.S. Patent No. 10,764,874 asserted by Petitioner.

C. Lead and Back-Up Counsel (37 C.F.R. § 42.8(b)(3))

Lead Counsel	Back-Up Counsel
Mark Garrett, Reg. No. 44,699	Richard S. Zembek, Reg. No. 43,306 David Ben-Meir, Reg. No. 46,152 Darren Smith, Reg. No. 64,261 Jeremy Albright, Reg. No. 73,379

D. Service Information (37 C.F.R. § 42.8(b)(4))

Lead Counsel	Back-Up Counsel
Mark T. Garrett Norton Rose Fulbright US LLP 98 San Jacinto Boulevard, Suite 1100 Austin, Texas 78701 512.536.3031 (telephone) 512.536.4598 (facsimile) mark.garrett@nortonrosefulbright.com	Richard S. Zembek Darren Smith Norton Rose Fulbright US LLP 1301 McKinney, Suite 5100 Houston, Texas 77010 richard.zembek@nortonrosefulbright.com 713.651.5283 (telephone) darren.smith@nortonrosefulbright.com 713.651.8361 (telephone) David H. Ben-Meir Norton Rose Fulbright US LLP 555 South Flower Street Los Angeles, California 90071 213.892.9202 (telephone) 213.892.9494 (facsimile) david.ben-meir@nortonrosefulbright.com Jeremy Albright Norton Rose Fulbright US LLP 98 San Jacinto Boulevard, Suite 1100 Austin, Texas 78701 512.536.3111 (telephone) 512.536.4598 (facsimile) jeremy.albright@nortonrosefulbright.com

Petitioner consents to electronic service.

III. GROUNDS FOR STANDING

Pursuant to 37 C.F.R. § 42.104(a), Petitioner certifies that the '037 Patent is available for *inter partes* review, and Petitioner is not barred or estopped from requesting an *inter partes* review on the grounds identified below.

IV. STATEMENT OF PRECISE RELIEF REQUESTED FOR EACH CLAIM CHALLENGED

A. Claims for Which Review Is Requested: 1-3, 5-6, 19-21

B. Statutory Grounds of Challenge (37 C.F.R. § 42.104(b)(2))¹

Claims 1-3, 5-6, and 19-21 are invalid under § 103(a) over the following prior art and combinations of prior art:

Ground	Claims	Prior Art
1	1-3, 5-6, 19-21	Smith
2	20	Smith, Bolotin
3	19-21	Smith, Little
4	20	Smith, Bolotin, Little
5	5-6	Smith, Challenger

Filed January 7, 2007, Smith is prior art under §102(e). Published February 1, 2007, Bolotin is prior art under § 102(b). Published February 27, 2003, Challenger is prior art under § 102(b). Published May 25, 2006, Little is prior art under §102(b). Filed March, 24, 2008, Dyer, which is used to reflect a POSITA's general

¹ All Grounds include a person of ordinary skill in the art's ("POSITA's") general knowledge at the time of the alleged invention ("TOI," which is treated as April 30, 2008, the '037 Patent's earliest-claimed priority date ("ECPD")), which is evidenced by the prior art, including that which is cited in support of, but not explicitly listed as part of, the grounds. *Koninklijke Philips N.V. v. Google LLC*, 948 F.3d 1330, 1337-38 (Fed. Cir. 2020).

knowledge, is prior art under § 102(e). Faddell, published November 11, 2004 and used to reflect a POSITA's general knowledge, is prior art under § 102(b). Stevens, published December 16, 2003 and used to reflect a POSITA's general knowledge, is prior art under § 102(b).

Petković further reflects the general knowledge of a POSITA. Madisetti, ¶ 33. Though not required to serve that purpose, Petković was also publicly-available prior to the TOI and is prior art under § 102(a) as evidenced by: (1) Petković's 2007 copyright date (Petković, 4; Madisetti, ¶ 33); (2) Špánek's February 2008 citation to Petković (Špánek, 1, 24, 125); (3) OECD's citation to Petković in January 2008 (OECD, 1-3, 18, 21, 67); and (4) Petković being available for purchase through Amazon by November 1, 2007 (Amazon, 1-2; Archive, ¶¶ 1-7 (¶¶ 5-6: explaining URL and attachment); Madisetti, ¶ 33 (explaining the November 1, 2007 URL of Amazon with reference to Archive, ¶¶ 5-6).

V. FIELD OF TECHNOLOGY²

The '037 Patent describes mechanisms for establishing a secure computer system, i.e., a computer system of enhanced integrity and security, and, more specifically, mechanisms for securely booting a computer system based on the use

² This explanation is from the perspective of a POSITA as of the ECPD. Madisetti, ¶ 32.

of an encryption key and pre-boot instructions, both of which are stored in the computer system's secure read-only memory ("ROM"). '037 Patent, Abstract, 1:8-12; Madisetti, ¶ 32.

A. Secure Hardware

Security of a computer system is typically established in the hardware of the computer system itself. Petković, 120;³ Madisetti, ¶ 34. Some portion of the hardware is immutable or "trusted," enabling security or "trust" in the whole platform. Petković, 120; Madisetti, ¶ 34. That trusted hardware can be leveraged either by delegating all critical functions to the trusted hardware, or by using the trusted hardware to extend a trusted platform in software. Petković, 121-122; Madisetti, ¶ 34. For the latter case, a secure non-volatile memory is used to ensure the security of the computer system's boot process, where a non-modifiable part of the BIOS (basic input/output system)⁴ is stored and executed before any other code. Petković, 121-122; Madisetti, ¶ 34. The secure memory may also be used to maintain keys or to securely store the signatures that the code components are compared against, allowing a secure update of system components. Petković, 122, 124; Madisetti, ¶ 34.

³ All Petković cites are to Petković's native page numbers. Madisetti, ¶ 34 n.1.

⁴ '037 Patent, 2:38-43.

B. Secure Boot

Ensuring that the code running on a computer system is indeed the code that is expected to run is essential to providing a secure computer platform. Petkovic, 123; Madisetti, ¶ 35. In a system where trust must be extended outside the trusted hardware onto a general use platform, boot protection, i.e., preventing unverified code configurations from being executed, is necessary. Petkovic, 123; Madisetti, ¶ 35. To accomplish this objective, the system needs a trusted, non-modifiable basic boot block, i.e., the very first code that is executed when the platform is powered on, which is usually part of the BIOS ROM. Petkovic, 123; Madisetti, ¶ 35. The BIOS verifies the next piece of code and then hands over control to that code, thereby extending the chain of trust. Petkovic, 123; Madisetti, ¶ 35. In such an implementation, every component called in the boot sequence can verify a hash or a digital signature of its successor and abort if an error is detected. Petković, 124; Madisetti, ¶ 35.

VI. LEVEL OF ORDINARY SKILL IN THE ART

A POSITA as of the ECPD⁵ would have had at least a bachelor's degree in electrical engineering, computer engineering, computer science, or a related degree,

⁵ All petition statements about the knowledge and skills of, and what would have been disclosed or obvious to, a POSITA are from this perspective as of this date.

and at least 2-3 years of experience in processor architecture, computer security, or a related field. Madisetti, ¶¶ 36-38, 23-24. The prior art and the '037 Patent also evidence this level of ordinary skill. *See Chore-Time Equip., Inc. v. Cumberland Corp.*, 713 F.2d 774, 779 (Fed. Cir. 1983); *Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001).

VII. THE '037 PATENT

Entitled “Computer System Comprising a Secure Boot Mechanism on the Basis of Symmetric Key Encryption,” the '037 Patent's secure boot process provides for executable instructions and data values that are accessed by a CPU in a way that cannot be circumvented so that a defined root for initialization of the platform is obtained. '037 Patent, 3:11-19; Madisetti, ¶ 39.

A. Background and Perceived Shortcomings

The '037 Patent confirms that “a plurality of encryption techniques are available, such as symmetric and asymmetric encryption/decryption techniques which enable exchange of data with a high degree of resistance against tampering.” '037 Patent, 1:46-51; Madisetti, ¶ 41. The '037 Patent also confirms that “a secure

See Madisetti, ¶ 30; *see also id.*, ¶¶ 1-3 (scope of work), 4-20 (background/qualifications), 21-29 (compensation, legal understanding, and basis for opinions).

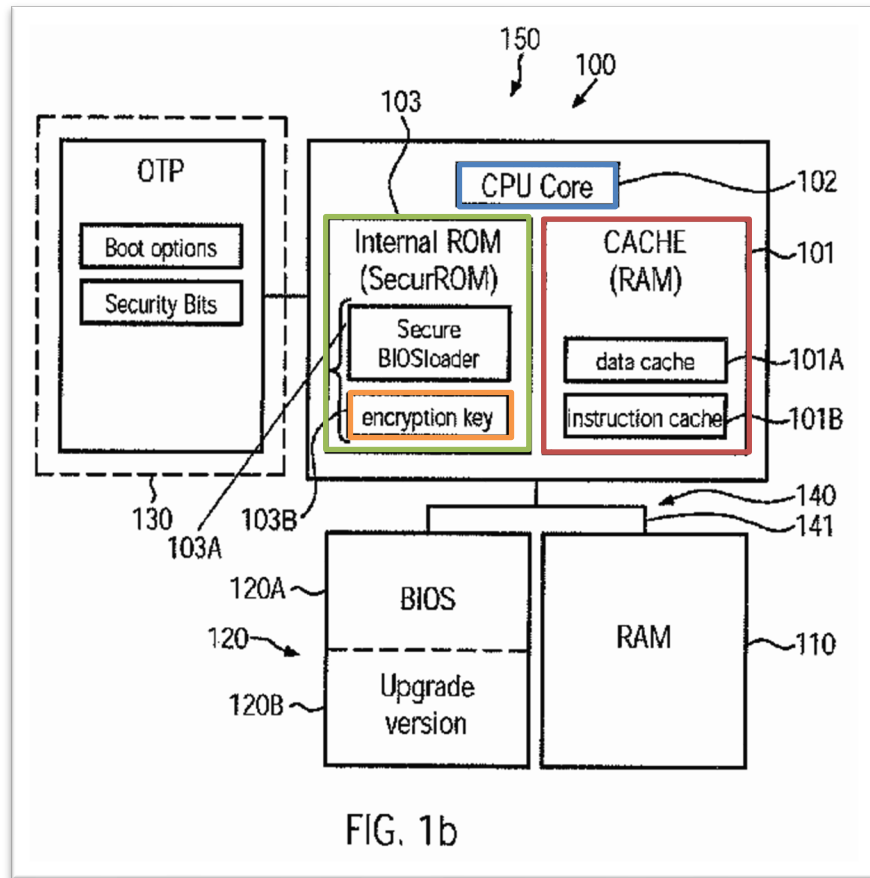
platform configuration may be established only by [sic, by] establishing a chain of trusted platform levels, where each subsequent level may be initialized by the preceding level only if the integrity of the subsequent level has been verified.” ’037 Patent, 2:1-5; *see* Madisetti, ¶ 40. The ’037 Patent states that “in a typical computer platform,” “the process of initializing the various hardware components of the computer system, such as the system memory and the execution of the instructions contained therein for initializing the higher system levels, which may also be referred to as boot strapping or booting, may have to be incorporated into the chain of trust to provide enhanced overall system integrity.” ’037 Patent, 2:13-33; *see* Madisetti, ¶ 41.

The ’037 Patent then describes “a typical start sequence for initializing a computer system after applying power or after a reset event.” ’037 Patent, 2:34-38; Madisetti, ¶ 42. The sequence begins with a “‘power on self-test’ of the CPU.” ’037 Patent, 2:34-38; *see* Madisetti, ¶ 42. After the self-test starts, and under the control of instructions frequently referred to as BIOS, the remaining self-test may be performed and the hardware configuration of the platform may be determined. ’037 Patent, 2:34-46; Madisetti, ¶ 42. Then, a search for a bootable device is performed, from which a primary boot block maybe loaded into system memory, with system control thereafter passed to the primary boot block. ’037 Patent, 2:46-52; Madisetti, ¶ 42.

The '037 Patent states that “[a]lthough great efforts have been made [to enhance] security of the entire boot process, it is difficult to provide a core root of trust for measurement (CRTM) for the boot process, for example, in view of platform modifications such as replacement of the non-volatile memory chip including the BIOS software. '037 Patent, 2:53-60; *see* Madisetti, ¶ 43. The '037 Patent purports to improve over the prior art by providing both a trust center for the secure boot process and flexibility for updating boot code. '037 Patent, 2:61-64; *see* Madisetti, ¶ 43.

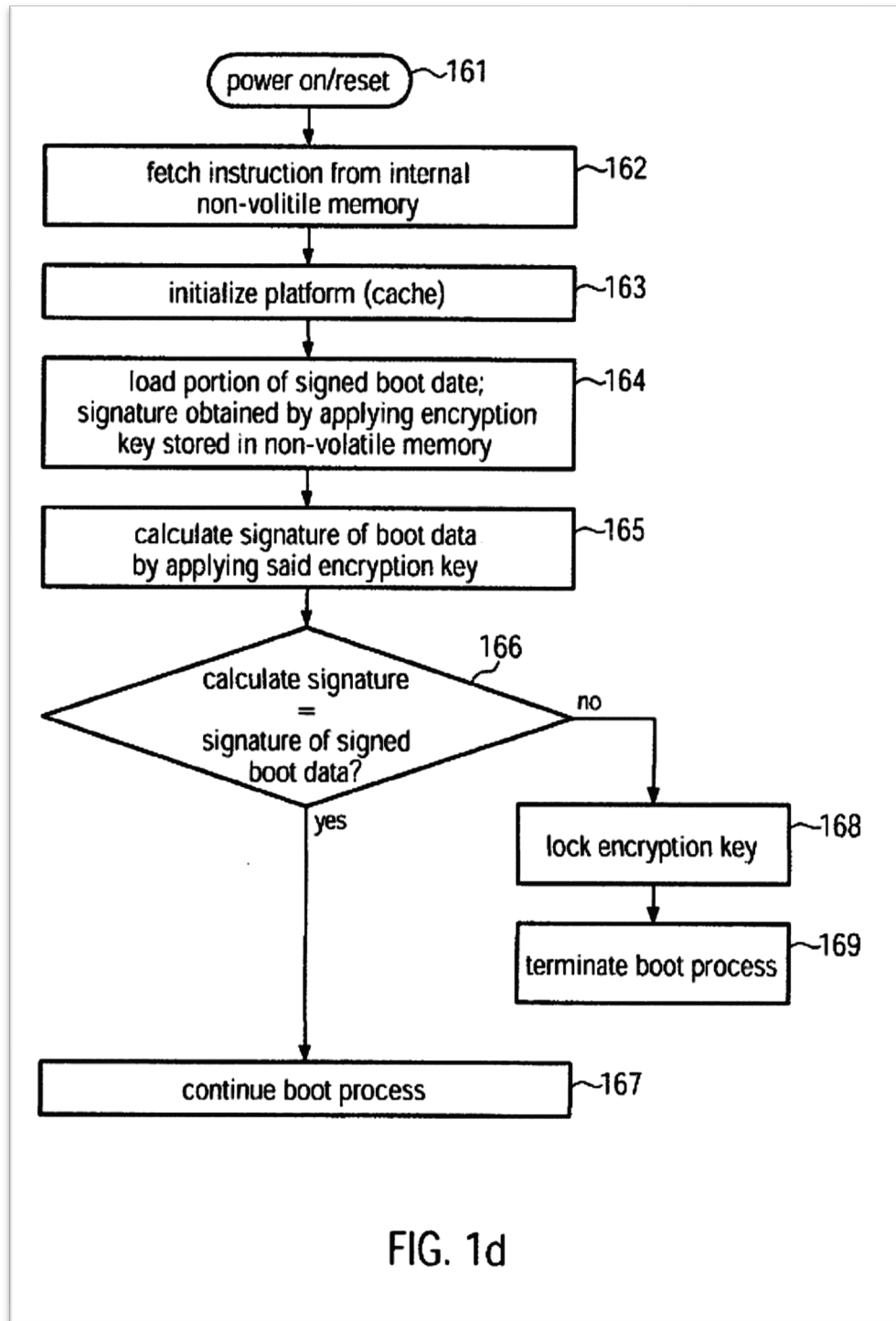
B. The '037 Patent's Asserted Improvement

The '037 Patent discloses a computer system that provides enhanced security in the boot process. '037 Patent, 1:8-12; Madisetti, ¶ 32. As annotated below, in the '037 Patent's FIG. 1b, computer system 150 includes a central processing unit (CPU) 100 that contains a CPU core 102 (outlined in **blue**), a secure (lockable) ROM 103 (outlined in **green**) and RAM in the form of a cache 101 (outlined in **red**). '037 Patent, 8:41-49; Madisetti, ¶ 49.



The ROM 103 contains a data set 103A in the form of a secure BIOS loader containing pre-boot instructions and data values, and a symmetric encryption key 103B (outlined in **orange**). '037 Patent, 7:35-41; Madisetti, ¶ 50. The symmetric encryption key is used to sign and verify a portion of boot information before it is used by the CPU. '037 Patent, 7:41-45; Madisetti, ¶ 50.

The '037 Patent's FIG. 1d reproduced below depicts a process for verifying a portion of a boot routine based on verification of the boot routine's signature. '037 Patent, 10:38-41; Madisetti, ¶ 51.



The '037 Patent indicates that a hash value of a first boot routine portion has initially been created to generate and store a signature 121S using the symmetric

encryption key 103B. '037 Patent, 11:42-52; *see* Madisetti, ¶ 52. In the verification process, the signed boot data portion 121A is loaded into RAM. '037 Patent, 11:42-52; *see* Madisetti, ¶ 52. Then, a signature of the portion 121A “is generated by, for instance, calculating a respective hash value ... and applying the encryption key 103B.” '037 Patent, 11:48-52; Madisetti, ¶ 52. The currently calculated signature is compared with the signature 121S, and if the values match, the boot process is permitted to continue. '037 Patent, 11:53-57; Madisetti, ¶ 52.

C. Prosecution History

See Section IX.A.

D. Claim Construction (37 C.F.R. § 42.104(b)(3))

Petitioner interprets the challenged claims of the '037 Patent according to the *Phillips* claim construction standard. 83 Fed. Reg. 51340, 51340-44 (Oct. 11, 2018); *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005).

Petitioner believes the term in Claim 20 reciting “**a locking mechanism configured to enable a portion of said non-volatile memory containing said symmetric encryption key upon one of a reset event and power-on event only**” should be construed in accordance with 35 U.S.C. § 112, ¶ 6. The absence of recited structure for the claimed locking mechanism rebuts the presumption that *locking mechanism* should not be construed as a means-plus-function term. *Williamson v. Citrix Online, LLC*, 792 F.3d 1339, 1348 (Fed. Cir. 2015).

The specification describes one structure for performing the claimed functions of locking and enabling a portion of the non-volatile memory containing the symmetric encryption key: lock mechanism 105. '037 Patent, 7:60-66; Madisetti, ¶¶ 124-125. The '037 Patent explains that the lock mechanism 105 may comprise “circuitry for shutting down supply energy for the memory [secure ROM] 103.” '037 Patent, 7:60-66; Madisetti, ¶ 125. And to “enabl[e]” the memory, a POSITA would have understood mechanism 105 to be able to provide power to the memory, at least in view of the '037 Patent's description that the lack of power to the memory renders the memory non-responsive. '037 Patent, 7:60-66; Madisetti, ¶ 125.

Therefore, a POSITA would have understood the Section 112, Paragraph 6 corresponding structure to be circuitry for enabling or disabling power to the secure ROM containing the symmetric encryption key. Madisetti, ¶ 125.

VIII. SPECIFIC EXPLANATION OF GROUNDS

A. Ground 1 – Claims 19-21, 1-3, and 4-5 Are Obvious Over Smith

1. Overview of Smith

Smith discloses a method and apparatus for secure booting of a device using codes stored in a secure ROM of the device and code images that are verified before they are executed. Smith, Title, Abstract; Madisetti, ¶ 61. Smith's FIGs. 1 and 2 depict the hardware configuration of the system (100) used for the secure booting. Smith, ¶¶ 12, 13, 30, 33, 38, FIGs. 1, 2; Madisetti, ¶ 61.

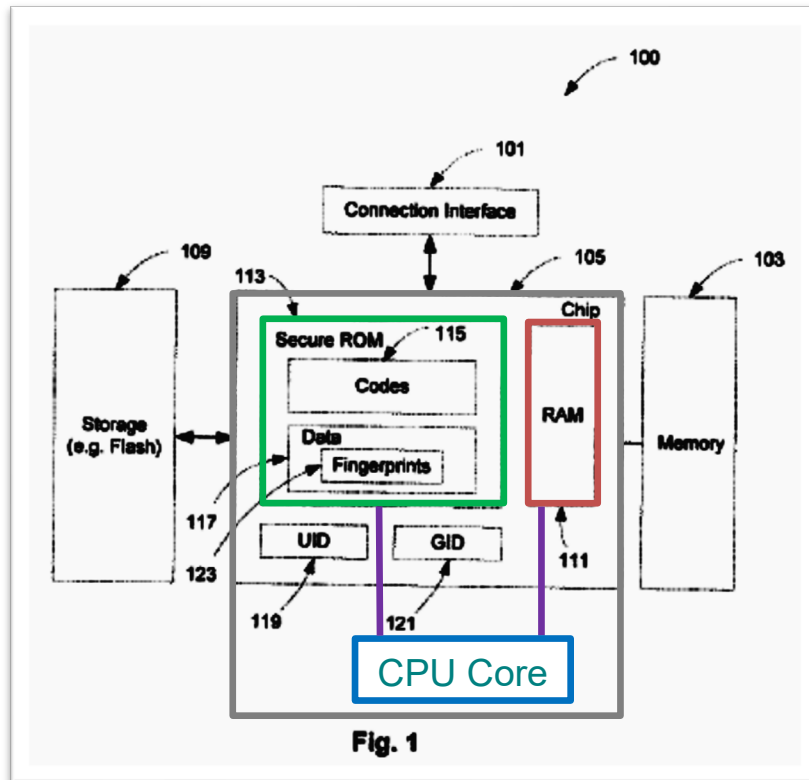
2. Element-by-Element Analysis

a) Limitations 19[pre] and 19[a]

As explained below, Smith's FIG. 1 chip 105 includes "a volatile random access memory" (**red**) and "a non-volatile memory" (**green**). Madisetti, ¶ 62. And it would have been obvious for that chip to further include:

- (1) a CPU core (**blue**) (discussed below in Section VIII.A.2.a)(3)) so that it could perform Smith's secure-booting processes as Smith describes and as the chip's EDRAM memory suggests, thereby rendering the chip "a central processing unit (CPU);" and
- (2) a bus system (**purple**) (discussed below in Section VIII.A.2.a)(4)) to allow the necessary communication between the CPU core and the memories.

See Madisetti, ¶ 62. Finally, Smith's "chip" 105 includes or obviously would have included a single substrate (**grey**) on which its disclosed and obvious components were formed (discussed below in Section VIII.A.2.a)(5)). *See* Madisetti, ¶ 62. These structures are shown on an annotated version of Smith's FIG. 1, reproduced below.
Id.



(1) **EDRAM 111 of chip 105 is a volatile random access memory**

Smith discloses that chip 105 includes a “RAM (Random Access Memory) component 111,” providing “EDRAM (Embedded Dynamic Random Access Memory)” as a specific example. Smith, ¶ 30, FIG. 1; Madisetti, ¶ 63. As a POSITA would have known, being a type of dynamic RAM, such EDRAM “requires power continually in order to refresh or maintain the data in the memory” (Smith, ¶ 64) and is therefore volatile. Microsoft, 558 (“**volatile memory** [is m]emory, such as RAM, that loses its data when the power is shut off”); Madisetti, ¶ 63 (including n.3).

(2) ROM 113 of chip 105 is a non-volatile memory

Smith's chip 105 further includes "a ROM 113" (Smith, ¶ 30), which is a "Read Only Memory" (*id.*, ¶ 28). Madisetti, ¶ 64. As a POSITA would have understood, ROM is a non-volatile memory. Microsoft, 367 ("**nonvolatile memory** [is a] storage system that does not lose data when power is removed from it" and is "[i]ntended to refer to ... ROM"), 333 ("memory storage can be ... *nonvolatile* and hold data permanently, such as ROM" (emphasis added)); Madisetti, ¶ 64.

(3) Chip 105 obviously would have included a CPU core

Smith's secure-booting processes involve the execution of codes, specifically, "executing codes embedded inside a device to verify a code image loaded in a memory of the device" before the code image is executed. Smith, Abstract; *see also*, *e.g.*, ¶¶ 7-9, FIG. 3 ("[e]xecut[ing] instructions in a ROM chip" to "[v]erify [a] code image," and, if so, "[e]xecut[ing] the code image to perform booting operations"), FIG. 4 and ¶ 42 (executing a "Secure Hash Algorithm" and an "Advanced Encryption Standard[] algorithm" as part of verifying a code image), FIG. 6 ("[r]eceiving an instruction ... to execute the code image"), FIG. 8 (same as well as "executing the trusted kernelcache" and "[r]eceiving and executing command calls"), FIG. 9 ("[e]xecuting the verified code image"), FIG. 10 (similar); Madisetti, ¶ 65. Executing such codes, as a POSITA would have understood, was performed using a CPU core. Madisetti, ¶ 65. Smith explains as much, disclosing that "a

microcontroller or other form of *processing core*”—i.e., a CPU core—is what “*executes program code instructions.*” Smith, ¶ 71 (emphasis added); Madisetti, ¶ 65.

It would have been obvious to include a CPU core like Smith describes in Smith’s chip 105. *See* Madisetti, ¶ 66. First, Smith explains that his secure-booting processes are performable with his FIG. 1 system 100. *See* Smith, ¶ 30 (“FIG. 1 ... illustrat[es] ... system components *for secure booting*” (emphasis added)); Madisetti, ¶ 66. With respect to Smith’s FIGs. 3 and 4 processes, Smith makes this explicit: those processes “may be performed by system 100 of FIG. 1.” Smith, ¶ 38 (FIG. 3); *see also* Smith, ¶ 42 (FIG. 4); Madisetti, ¶ 66. With the understanding that these secure-booting processes were performed using a CPU core (*supra*), Smith’s explanation that system 100 is capable of performing them would have both suggested the inclusion of a CPU core in that system and motivated such inclusion to enable its Smith-described capability. Madisetti, ¶ 66. In the single-chip version of Smith’s system 100, this obvious-to-include CPU core would have been on Smith’s chip 105. Smith, ¶ 30 (“System 100 may reside in *one* or more chips inside a device.” (emphasis added)); Madisetti, ¶ 66.

Second, as a POSITA would have understood, Smith’s on-chip EDRAM 111 (*see* Section VIII.A.2.a)(1)) is “a capacitor-based dynamic random access memory usually *integrated on the same die or in the same package as the main ASIC or*

processor.” Dyer, ¶ 28 (emphasis added); Madisetti, ¶ 67. Such presence of EDRAM 111 on chip 105 would have therefore at least suggested to a POSITA the further presence of a CPU core on the chip with which the EDRAM was integrated, particularly in view of their understanding that the secure-booting processes performable with Smith’s FIG. 1 system 100 including the chip involved the use of a CPU core (*supra*). Madisetti, ¶ 67.

(4) Chip 105 obviously would have included a bus system

A POSITA would have understood that the obvious CPU core of Smith’s chip 105 (*see* Section VIII.A.2.a)(3)) needed to communicate with the volatile memory (*see* Section VIII.A.2.a)(1)) and non-volatile memory (*see* Section VIII.A.2.a)(2)) of the chip in order to perform Smith’s secure-boot processes. Madisetti, ¶ 68 Smith’s FIG. 3 secure-boot process, for example, is explicitly performable by Smith’s FIG. 1 system 100 and includes “[e]xecut[ing] instructions in a ROM chip to locate a code image for booting a device,” “[l]oad[ing] the code image into a memory,” and “[e]xecut[ing] the code image” if it is verified. Smith, FIG. 3; *see also id.*, ¶¶ 38-40; Madisetti, ¶ 68. Those code-image-locating instructions “may be read from a code section of the ROM chip as in codes 115 of FIG. 1” (Smith, ¶ 38)—i.e., may reside in Smith’s ROM 113 (Smith, ¶ 30, FIG. 1). Madisetti, ¶ 68. And the code image may be loaded into Smith’s RAM 111 for post-verification execution. Smith, ¶¶ 38, 40 (“the [executed] code image may be LLB, an iBoot or

a Kernelcache as shown in 225, 227 and 223), FIG. 2 (showing LLB and iBoot in Smith's RAM 111), ¶ 31 ("codes 115 may cause loading a code image into a device memory such as ... RAM 111"); Madisetti, ¶ 68.

A POSITA also knew how to achieve such communication between the CPU core, RAM 111, and ROM 113 of Smith's chip 105: via a bus. Microsoft, 77-78 (a "bus [is a] set of hardware lines (conductors) used for data transfer among the components of a computer system"); Madisetti, ¶ 69. As Smith explains with respect to his FIG. 12 data processing system 1200 that "may be implemented as part of system [100] as shown in FIG. 1" and includes, akin to Smith's chip 105, a "memory 1201 for storing ... programs for execution by the processing system" and "a processing system 1211, which may be one or more microprocessors" (Smith, ¶ 66), "[i]t will be appreciated that one or more buses ... may be used to interconnect the various components as is well known in the art" (Smith, ¶ 68). Madisetti, ¶ 69; *see also* Smith, ¶¶ 64, 65, 72, FIG. 11 (describing and depicting buses as "well-known" structures for interconnecting components).

It would have been obvious to include a bus like Smith describes on Smith's chip 105 because doing so would have provided for the above-described communication between the CPU core, RAM 111, and ROM 113 and thereby enabled chip 105 to perform the above-described secure-boot processes. *See* Madisetti, ¶ 70. Specifically, and as reflected in **purple** in the annotated version of

Smith's FIG. 1, reproduced above, that bus would have included a portion connecting the CPU core to ROM 113 so that the CPU core could read and execute the code-image-locating instructions contained therein, as well as a portion connecting the CPU core to RAM 111 so that the CPU core could read and execute the code image loaded therein. *Supra*; Madisetti, ¶ 70. Such a bus satisfies the claimed "bus system." Madisetti, ¶ 70.

(5) "Chip" 105 included volatile memory and non-volatile memory as well as, obviously, a processor core and a bus system, all of which would have therefore been on a substrate

As a POSITA would have understood, the volatile and non-volatile memories, CPU core, and bus system of Smith's "chip" 105 would have been "fabricated on a single chip of ... semiconductor material." Microsoft, 98, 277 (a "**chip**" is an "integrated circuit," which is "[a] device consisting of a number of connected circuit elements ... fabricated on a single chip of ... semiconductor material"); Madisetti, ¶ 71. Such chip of semiconductor material corresponds to the claimed substrate. Madisetti, ¶ 71.

b) Limitation 19[b]

Smith discloses this limitation. Madisetti, ¶ 72.

(1) Certain codes 115

Several of Smith's "codes 115" qualify as the claimed "pre-boot information." Madisetti, ¶ 73. For example, Smith describes "codes 115" that "may cause

hardware initialization for the device to support a connection or communication interface such as USB (Universal Serial Bus) ... [and] include instructions to change the clock rate of the device” (hereinafter, “First Codes 115”). Smith, ¶ 30; Madisetti, ¶ 74. First Codes 115 qualify as the claimed “pre-boot information” because, as a POSITA would have understood based on Smith, their execution: (1) would have caused one or more activities (the referenced hardware initialization and the referenced clock rate change) that preceded the code image execution that Smith describes as part of process 300 for secure booting (*see* Smith, ¶¶ 38-41) and that Smith describes and illustrates as being for the purpose of performing “booting operations” (*see id.*, ¶ 40, FIG. 3 (block 311: “Execute the code image to perform booting operations”), and/or (2) would not have been changed or interrupted, which the ’037 Patent indicates qualifies them as “pre-boot” information (*see* ’037 Patent, 3:21-25). Madisetti, ¶ 74.

Smith also describes “codes 115” as including instructions that, when executed, locate a code image stored in a storage component of the device in which chip 105 was positioned (hereinafter, “Second Codes 115”). Smith, ¶ 38; Madisetti, ¶ 75. Second Codes 115 also qualify as the claimed “pre-boot information” because, as a POSITA would have understood based on Smith, their execution: (1) would have caused the referenced code image location (*see* Smith FIG. 3 at blocks 301 and 319) before booting operations started (*see* Smith FIG. 3 at block 311 and Smith ¶

40 (describing “booting operations” starting with the execution of the code image)), and/or (2) would not have been changed or interrupted, which the ’037 Patent indicates qualifies them as “pre-boot” information (*see* ’037 Patent, 3:21-25). Madisetti, ¶ 75.

Second, Smith discloses that ROM 113 (i.e., the claimed “non-volatile memory” as explained in Section VIII.A.2.a)(2)) “stores the codes 115.” Smith, ¶¶ 30, 38 (explaining the process therein described (process 300) is performed with system 100 and that the instructions of the Second Codes 115 are “in a ROM chip,” which a POSITA would have understood to refer to ROM 113 in which codes 115 are depicted as stored in FIG. 1); Madisetti, ¶ 76. Accordingly, First Codes 115 and Second Codes 115 qualify as the claimed “pre-boot information stored in non-volatile memory.” Madisetti, ¶ 76.

Next, as a POSITA would have understood, the hardware initialization and/or the clock rate change caused by First Codes 115 would necessarily have been caused by instructions corresponding to those codes being executed by the CPU core of chip 105. *See* Smith, ¶ 30 (“Codes 115 may include *instructions* to change the clock rate of the device.” (emphasis added)); Madisetti, ¶ 77. And, as a POSITA would have understood, the code image location caused by Second Codes 115 also would have necessarily been caused by instructions corresponding to those codes being executed by the CPU core of chip 105. *See* Smith, ¶ 38 (“the processing logic of process 300

may locate a code image by executing *instructions* in a ROM chip at block 301 ... read from a code section of the ROM chip as in codes 115 of FIG. 1” (emphasis added)); Madisetti, ¶ 77. Therefore, both First Codes 115 and Second Codes 115 include “instructions executable by [the] CPU core,” as claimed. Madisetti, ¶ 77.

(2) UID 119

Further, Smith discloses “pre-boot information including ... a symmetric encryption key.” Madisetti, ¶ 78. Smith discloses that UID 119 for a device that system 100 is securely booting is a symmetric encryption key for that device. *See* Smith, FIG. 2, ¶¶ 28, 32 (UID 119 is unique for the device); Madisetti, ¶ 78.

First, UID 119 qualifies as “pre-boot information” because, as a POSITA would have understood based on Smith, its use: (1) would have been considered by a POSITA as “pre-boot” (and, thus, UID 119 as “pre-boot information”) because such use preceded the code image execution (block 311) of process 300 that Smith describes in FIG. 3 as performing “booting operations” (*see* Smith, ¶ 40, FIG. 3 (block 311 text), and/or (2) would not have been changed or interrupted, which the ’037 Patent indicates qualifies it as “pre-boot” information (*see* ’037 Patent, 3:21-25). Madisetti, ¶ 79.

Second, UID 119 is a symmetric encryption key. Madisetti, ¶ 80. Smith discloses that the UID may be used to create a signature value over a code image “through well-known hashing and encryption algorithms.” Smith, ¶ 39; Madisetti,

¶ 80. The same UID is used to generate a current signature—described as deriving another signature—for the code image using the same well-known hashing and encryption algorithms. Smith, ¶ 39 (describing block 305), FIG. 3 (depicting block 305); Madisetti, ¶ 80. Elsewhere, Smith describes that current-signature generation with reference to process 400 shown in FIG. 4 as “perform[ing] an encrypting operation at block 403 to generate a signature 405 based on a key string 413, UID 401 and seed string 407 associated with a device.” Smith, ¶ 42, FIG. 4; Madisetti, ¶ 80. A POSITA would have understood UID 119 corresponds to UID 401 because Smith states, “UID 401 may be stored within the device as UID 119 shown in FIG. 1.” Smith, ¶ 42, FIG. 4; Madisetti, ¶ 80. Smith also states the encrypting operation can be “based on AES (Advanced Encryption Standard) algorithm.” Smith, ¶ 42, FIG. 4; Madisetti, ¶ 80. Because AES is an algorithm for symmetric key encryption, a POSITA would have understood the UID to be a symmetric encryption key. Madisetti, ¶ 80.⁶ Indeed, Smith confirms, “the verification of the code image is based on a key stored within the secure storage area. The *key may be unique to each device.*” Smith, ¶ 7 (emphasis added); *see also id.*, ¶ 8 (“Upon being successfully

⁶ Hereinafter, such current-signature generation described in this paragraph with reference to Smith paragraphs 39 and 42 and FIGs. 3-4 shall be referred to as Smith’s “Current Signature Generation.” Madisetti, ¶ 80 n.4.

verified using a key stored within the secure ROM and *the key uniquely identifying the device ...*” (emphasis added)), claim 5 (“wherein the key [stored within the ROM] is configured to uniquely identify the device”), claim 13 (same), claim 19 (describing a computer implemented method involving “a key stored within the secure ROM [that] uniquely identif[ies] the portable device”), claim 22 (describing machine readable media that, when executed by a machine, perform a method that involves “a key stored within the secure ROM [that] uniquely identif[ies] the portable device”), claims 25 and 28 (similar to claims 19 and 22, respectively, except referencing a “unique identifier (ID)” rather than “key”); Madisetti, ¶ 80.

As confirmation, the '037 Patent itself describes that the symmetric encryption key may be in the form of an identifier that is unique to a computer system. Madisetti, ¶ 81. The '037 Patent states, “[i]n symmetric key encryption techniques, a unique key is used for both the encryption and decryption of data, wherein access to the unique key may have to restricted to authorized parties.” '037 Patent, 1:55-58; Madisetti, ¶ 81. The '037 Patent also states that the unique symmetric encryption key may be unique to each computer system, like the UID in Smith, so as to not undermine the security of a computer system when another computer system’s symmetric key has been compromised. *See* '037 Patent, 6:61-66; Madisetti, ¶ 81. Going further, the '037 Patent explains that the symmetric encryption key may be determined and—like Smith’s UID (Smith, ¶ 28)—

incorporated into the CPU during manufacturing: “the degree of security may be determined on the basis of the granularity of the symmetric encryption key, which may be incorporated into the non-volatile memory during fabrication of the CPU device.” ’037 Patent, 14:26-28; Madisetti, ¶ 81.

Accordingly, Smith discloses pre-boot information including a symmetric encryption key. Madisetti, ¶ 82.

Third, Smith discloses that UID 119 is stored in secure ROM 113. Madisetti, ¶ 83. As Smith states:

[T]he security of booting a device may be performed by the code and data stored inside a secure storage area such as a ROM (Read Only Memory), also referred to as a secure ROM, integrated together within the device. The content of a secure ROM may be stored during a manufacturing stage of the device. The secure ROM may be associated with a UID (Unique Identifier) of the device.

Smith, ¶ 28; Madisetti, ¶ 83. Here, Smith describes a single secure ROM that contains the code and data for performing the secure booting processes of the device. Smith, ¶ 28; Madisetti, ¶ 83. This passage then references the “content” of the secure ROM, and then specifically identifies the UID as an instance of that content in the secure ROM. Smith, ¶ 28; Madisetti, ¶ 83. In system 100 depicted in FIGs. 1 and 2—the only system Smith discloses for performing Smith’s secure booting processes—secure ROM 113 is the only secure ROM that is depicted. Smith, FIGs.

1, 2; Madisetti, ¶ 83. Further, in the context of the process described in FIG. 6, which Smith explains is performable with system 100 from FIG. 1, Smith recites regarding the UID's secure storage location: "the processing logic of process 600 may derive a hash signature from the code image based on a UID stored in a secure ROM of the device, such as UID 119 as shown in FIG. 1." Smith, ¶ 47; Madisetti, ¶ 83. Similarly, Smith recites, "[t]he device may perform a chain of certifications to verify the received code image is trusted based on a UID and a fingerprint stored inside a secure ROM of the device such as UID 119 and fingerprints 123 of FIG. 1." Smith, ¶ 47; Madisetti, ¶ 83. Smith paragraphs 49, 51, 53, and 61 contain additional, similar disclosures. Madisetti, ¶ 83. In the "Summary of the Description" section, Smith states the "unique identifier (ID) uniquely identifying the device"—which a POSITA would understand to be the UID Smith references elsewhere—can be stored in the secure ROM." Smith, ¶ 9; Madisetti, ¶ 83. Smith's claims 5, 19, 22, 25, and 28 provide similar support, as explained above. Madisetti, ¶ 83. In view of these Smith disclosures, and because the only secure ROM depicted in FIGs. 1 and 2 or otherwise referred to in Smith is secure ROM 113, a POSITA would have understood that the secure ROM storing UID 119 is secure ROM 113. Madisetti, ¶ 83.

Smith also discloses that UID 119 is "for verifying at least a portion of a boot routine signed by using said symmetric encryption key." Madisetti, ¶ 84. First, Smith expressly discloses a process that "may verify whether the loaded code image

could be trusted based on a UID associated with the device such as UID 119 of FIG. 1.” Smith, ¶ 39, FIGs. 1 and 3; Madisetti, ¶ 84. Smith also discloses that the loaded code is a boot routine, namely the low level boot (“LLB”) code image 229 depicted in FIG. 2. Smith, FIGs. 2 and 3, ¶ 40; Madisetti, ¶ 84. LLB 229 is a boot routine that is executed after it is loaded into RAM 111 as LLB 225. Smith, FIGs. 2 and 3, ¶¶ 40, 33-34; Madisetti, ¶ 84. Thus, Smith discloses using UID 119 to verify (at least a portion of) a boot routine. Madisetti, ¶ 84.

Second, Smith discloses the boot routine to be verified is signed using UID 119. Madisetti, ¶ 85. Specifically, Smith discloses that the loaded code image being verified contains a header value that “may include a signature value signed over the code image according to the UID of the device through well-known hashing and encryption algorithms.” Smith, ¶ 39; Madisetti, ¶ 85. To the extent Patent Owner (“PO”) argues more than the foregoing is required, Smith’s Current Signature Generation, which applies to boot routine LLB 225 (Smith, ¶ 40, FIG. 2), also meets this aspect of Limitation 19[b]. Madisetti, ¶ 85.

Smith therefore discloses a symmetric encryption key in the form of UID 119 for verifying at least a portion of a boot routine signed by using the symmetric encryption key. Madisetti, ¶ 86.

c) Claim 20

If the claim term “locking mechanism configured to enable a portion of said non-volatile memory containing said symmetric encryption key” is not construed as a means-plus-function term, then Smith discloses the claimed locking mechanism. *See* Madisetti, ¶¶ 87-92.

Smith discloses processing logic in the form of a trusted code image that disables access to or locks UID 119 (i.e., the claimed symmetric encryption key) in secure ROM 113. Smith, ¶ 61; Madisetti, ¶ 87. Specifically, Smith explains that “the processing logic of process 1000[, performable by system 100 for executing unverified code image,] may disable accessing to a UID of a secure ROM in a device such as UID 119 in FIG. 1. In one embodiment, a trusted code image may be configured to turn off accessing to the UID when executed.” Smith, ¶ 61, FIGs. 1, 2, 10; Madisetti, ¶ 87. Smith also recites that the “trusted code image may be a boot image verified by codes inside a secure ROM ... such as codes 115 in FIG. 1.” Smith, ¶ 61; Madisetti, ¶ 87. Smith also discloses a hardware switch configured to disable access to UID 119 of secure ROM 113. Smith, ¶ 61; Madisetti, ¶ 87. Accordingly, Smith discloses that, in a process performed by system 100 of FIGs. 1 and 2, UID 119 in secure ROM 113 may be locked from access. Madisetti, ¶ 87.

Smith also discloses to a POSITA that ROM 113 is accessible or enabled before UID 119 is used following a reset or reboot of the system. Madisetti, ¶ 88.

For example, Smith discusses accessing the UID when codes 115 in ROM 113 are verifying a first code image, like LLB 225, using the UID. Smith, ¶ 32; Madisetti, ¶ 88. As a POSITA would have understood, it would not have been possible to access UID 119 in ROM 113 if at least the portion of ROM 113 where UID 119 was stored had not earlier been enabled. Madisetti, ¶ 88.

To a POSITA, these Smith disclosures describe a lock mechanism for disabling or enabling access to at least the portion of ROM 113 storing UID 119. Madisetti, ¶ 89.

Smith also discloses enabling the above-described portion of ROM 113 upon one of a reset event and power-on event only. Madisetti, ¶ 90. According to Smith, the unlocked (or enabled) state of ROM 113—when codes 115 are executed to access UID 119 and commence a secure boot process—corresponds to a reset event: “the processing logic of process 900 *may reset the device to reboot* from the operating system stored inside the device.” Smith, ¶ 60 (emphasis added); Madisetti, ¶ 90. Because a reset initiates a reboot of the system, and a reboot of the system is disclosed to enable access to the UID, Smith discloses to a POSITA enabling the above-described portion of ROM 113 “upon ... a reset event.” Madisetti, ¶ 90.

A POSITA also would have understood Smith to disclose enabling the above-described portion of ROM 113 upon a power-on event. Madisetti, ¶ 91. More specifically, a POSITA would have appreciated that a boot process for a computer,

including at least Smith's secure boot process 300, is associated with a power-up event in the same way that a reboot is associated with a reset event. *Id.* Furthermore, Smith does not disclose any access to UID 119 upon any event other than in the context of a system boot or reboot, which would necessarily follow a power-on event or reset event. *Id.* A POSITA would have also understood that to implement the functionality described in Smith for locking and unlocking access to the above-described portion of ROM 113 necessarily would have required something (such as circuitry and the CPU core executing instructions) to do so. *Id.*

Accordingly, Smith discloses "a locking mechanism configured to enable a portion of said non-volatile memory containing said symmetric encryption key upon one of a reset event or power-on event only." *Id.*, ¶ 92.

d) Limitation 21[pre]

To the extent limiting, Smith discloses claim 21's preamble. Smith's "invention relates to booting a computing device securely." Smith, ¶ 1; Madisetti, ¶ 93. The one example system that Smith provides for carrying out such secure booting is system 100, which is depicted in FIGs. 1 and 2. *See, e.g.,* Smith, FIGs. 1, 2, ¶¶, 30-32, 33 (describing FIG. 2's system 100 as "executing secure booting"), 38 (describing FIG. 3 as illustrating a process for performing securing booting using system 100), 42 (describing FIG. 4 as illustrating a process performable by system 100 to generate a signature from a code image), 45 (describing FIG. 6 as illustrating

a process to securely recover an operating environment from a host to a device performable by system 100); Madisetti, ¶ 93. Smith explains that system 100 resides in the device that is the subject of the secure booting. Smith, ¶¶ 30-32; Madisetti, ¶ 93. Smith then explains that some versions of the invention, which necessarily would have been understood to include system 100, could be part of a media player (like an Apple iPod®) that a POSITA would have understood qualified as a “computer system” because it had the basic components of a computer system: a storage device, a display device, an input device, and a processor. Smith, ¶¶ 69-71; *see also* Fadell (identified in Smith paragraph 70 as an example media player, and incorporated by reference in Smith), ¶¶ 44-47 (describing the media player’s components a POSITA would have understood reflected same qualified as a “computer system”); *see also* Smith, ¶¶ 66-68 and FIG. 12 (describing and depicting data processing system 1200 that “may be implemented as part of system [100] as shown in FIG. 1” and includes the basis components of a computer system); Madisetti, ¶ 93.

e) Limitation 21[a]

Smith’s computer system (identified in Sec. VIII.A.2.d)) includes chip 105, which is a central processing unit that satisfies Limitation 21[a] for the same reasons Limitation 19[pre] is satisfied. *See* Sec. VIII.A.2.a); Madisetti, ¶ 94.

f) Limitations 21[b][i]-21[b][ii]

The central processing unit that satisfies Limitation 21[a] meets Limitations 21[b][i] and 21[b][ii] for the same reasons Limitations 19[a] and 19[b], respectively, are satisfied. *See* Secs. VIII.A.2.a) and VIII.A.2.b); Madisetti, ¶ 95.

g) Limitation 21[c]

Smith discloses this limitation. Madisetti, ¶ 96. Smith's computer system (identified in Sec. VIII.A.2.d)) includes storage component 109 that takes the form of flash memory (Smith, ¶ 31, FIGs. 1 and 2), which a POSITA would have known was non-volatile memory. '037 Patent, 7:27-32 (indicating "flash memory" is a form of "non-volatile memory"); Madisetti, ¶ 96. Smith's flash memory form of storage component 109 is configured to hold the LLB boot routine, which as stored in storage component 109 is referred to as LLB 229, and a second boot routine in the form of iBoot 231. Madisetti, ¶ 96. Both LLB 225 and iBoot 231 are boot routines because each is code that is executed as part of a boot process. Smith, FIGs. 2, 3, ¶¶ 34, 40; Madisetti, ¶ 96. Thus, Smith's computer system includes the claimed "non-volatile boot memory." Madisetti, ¶ 96.

h) Limitation 1[pre]

To the extent limiting, Smith discloses claim 1's preamble. Smith discloses a method of booting a computer system, such as an Apple iPod®. *See* Smith, Title, FIG. 3, ¶ 38; Sec. VIII.A.2.d); Madisetti, ¶ 97.

i) Limitation 1[a]

Smith renders this limitation obvious. First, ROM 113 meets the claimed “internal non-volatile memory area of a central processing unit” for the same reasons set forth in Sections VIII.A.2.a) and VIII.A.2.a)(2) that ROM 113 qualifies as the “non-volatile memory” of claim 19. Madisetti, ¶ 98. To the extent PO argues otherwise, ROM 113 is an “internal” area of the CPU (chip 105) at least because ROM 113 is on the same substrate as the CPU’s CPU core. Sec. VIII.A.2.a)(5); Madisetti, ¶ 98.

Second, Smith renders obvious executing instructions stored in ROM 113 upon one of a power-on and a reset event, as recited in claim 1, which execution initializes RAM 111 (“an internal random access memory” as set forth in claim 2) and loads the LLB boot image (“a signed portion of boot data” as set forth in claim 1 and referred to in claim 2 as “said at least a portion of boot data”) from storage component 109 (“a non-volatile memory” as set forth in claim 2) into RAM 111. Madisetti, ¶ 99. The execution of the above-described instructions is addressed first (and referred to as the “Mapped Limitation 1[a] Instructions Execution”), followed by why such execution occurs upon one of a power-on and a reset event. *Id.*

Smith discloses that executing instructions in codes 115, stored in ROM 113, causes the loading of code image LLB from storage component 109 into RAM 111. Smith, ¶¶ 30 (codes 115 are stored in ROM 113), 31 (codes 115 cause loading a code

image into RAM 111), 32 (system 100, as reflected in FIG. 2, loads the LLB boot image (as LLB 229) from storage component 109 into RAM 111 (as LLB 225)), 38 (describing, as part of process 300 performed by system 100, executing instructions in codes 115 of “a code section of the ROM chip” (which a POSITA would have understood as located in ROM 113) to locate a code image “stored in a memory component” (which a POSITA would have understood as memory component 109) and load the located code image “into a memory” (which a POSITA would have understood as RAM 111)); Madisetti, ¶ 100.

As a POSITA would have understood, it would not have been possible to load the LLB code image into RAM 111 without first initializing RAM 111. *See* Microsoft, 273 (defining the verb “**initialize**” as “1. To prepare a storage medium, such as a disk or a tape, for use.”); Madisetti, ¶ 101. Furthermore, a POSITA would have understood codes 115 to accomplish such initialization at least because they are the first codes (and, thus, the first instructions) that are executed in Smith. *See* Smith, ¶ 38; Madisetti, ¶ 101. At a minimum, a POSITA would have found it obvious to make any modification to Smith’s system 100 (such as codes 115 thereof) required so that instructions of codes 115 to (when executed) initialized RAM 111 before loading the LLB code image into RAM 111 because: (1) as explained above, RAM 111 had to be initialized before such loading, otherwise such loading could not have taken place, and (2) it would have been logical to put such instructions in

codes 115 stored in ROM 113 because an on-chip ROM (like ROM 113) was where such instructions were typically stored (*see* Stevens, 2:30-61 (minimal amount of BIOS initialization code operative to initialize “the system memory” is stored in a first portion of “the critical nonvolatile storage device,” which can be ROM)). *See* Madisetti, ¶ 101.

Smith discloses that the Mapped Limitation 1[a] Instructions Execution occurs upon a power-on event because a POSITA would have understood the Mapped Limitation 1[a] Instructions Execution to necessarily occur after the computer system were turned on in order for the computer system to be usable for normal operation. Madisetti, ¶ 102. Smith renders obvious that the Mapped Limitation 1[a] Instructions Execution occurs upon a reset event because Smith discloses resetting the device (mapped to the “computer system” of the preamble) containing Smith’s system (100) following securely restoring software components from a host to the device (Smith, ¶ 58 (describing block 821 in FIG. 8: following receipt of a command to “restart the device” “the processing logic of process 800 may reset the device[, which then] reboot[s] from the operating system stored in the mass storage of the device”), FIG. 8 (showing block 821)) and following securely updating an application from a host to the device (*id.*, ¶¶ 20, 60 (describing block 911 in FIG. 9: “the processing logic of process 900 may reset the device to reboot

from the operating system stored inside the device”), FIG. 9 (showing block 911)).
See Madisetti, ¶¶ 103-104.

A POSITA would have understood that rebooting the device in each of these two circumstances would have involved either a warm reboot—which did not involve turning off power and turning power back—or a cold reboot—which did involve turning off power to the device and then turning on the power to the device. Microsoft, 442 (defining “**reboot**” as “[t]o restart a computer by reloading the operation system. *See also* boot², cold boot, warm boot.”), 107 (defining “**cold boot**” as “[a] startup process that begins with turning on the computer’s power.”), 561 (defining “**warm boot**” as “[t]he restarting of a running computer without first turning off the power.”). Madisetti, ¶ 103. Because Smith teaches rebooting (as just discussed), only two reboot options existed (warm and cold), both of which a POSITA would have been aware of, and both of which would have produced predictable results, a POSITA would have found it obvious to try a cold reboot of the device (*see* Madisetti, ¶ 104), rendering the same obvious (*see KSR*, 550 U.S. at 421). Because a cold reboot would have involved turning the power to the computer system off and then on, the Mapped Limitation 1[a] Instructions Execution would have necessarily occurred thereafter, just as it would have upon the power-on event discussed earlier. Madisetti, ¶ 104.

j) Limitation 1[b]

Smith discloses this limitation because ROM 113 contains UID 119 for the same reasons UID 119 is stored in ROM 113 set forth in Section VIII.A.2.b)(2), and UID 119 is an encryption key for the same reasons UID 119 is a symmetric encryption key in Section VIII.A.2.b)(2). Madisetti, ¶ 105.

k) Limitation 1[c]

Smith discloses this limitation. Madisetti, ¶ 106. The LLB code image, which is loaded into RAM 111 as LLB 225 (Smith, ¶ 33, FIG. 2), is at least a signed portion of boot data for which a current signature will be determined. Madisetti, ¶ 106. A POSITA would have understood the LLB code image as “boot” data because LLB stands for “Low Level Boot” (Smith, ¶ 33), the LLB code image is described elsewhere as a “boot image” (*see* Smith, ¶ 56 (describing receiving boot images in the FIG. 8 restore process), FIG. 8), and its execution involves “booting operations” (Smith, ¶ 40, FIG. 3 (block 311)). Madisetti, ¶ 106. Furthermore, due to the “at least” claim language, the LLB code image also qualifies as the claimed “boot data” because the LLB code image is at least a portion of itself. Madisetti, ¶ 106. Should PO disagree, the LLB code image is also at least a portion of itself and any one or more of the other code images stored in storage component 109, which a POSITA would have understood are also “boot” data for at least the latter two of the three reasons provided above for the LLB code image: the iBoot code image (231) and

the Kernelcache code image (233). Smith, FIG. 2, ¶¶ 33-35; Madisetti, ¶ 106. The LLB code image is signed because it includes a header value that includes “a signature value signed over the code image according to [UID 119] through well-known hashing and encryption algorithms.” Smith, ¶¶ 39 (explaining such signature value of a code image), 40 (reflecting that the LLB code image is a code image to which paragraph 39’s discussion applies); Madisetti, ¶ 106.

Next, Smith discloses “determining a current signature” of the loaded LLB code image. Madisetti, ¶ 107. In particular, Smith recites, “the processing logic of process 300 derives another signature value from the [LLB] code image according to [UID 119] through the same well-known hashing and encryption algorithms.” Smith, ¶ 39; *see also id.*, ¶ 42 (describing how UID 119 is used to determine a current signature of the loaded LLB code image), FIG. 4; Madisetti, ¶ 107. Smith’s deriving of another signature value from the LLB code image constitutes determining a current signature of the LLB code image loaded into RAM 111. Madisetti, ¶ 107.

I) Limitation 1[d]

Smith discloses this limitation. Madisetti, ¶ 108. The signature value signed over the LLB code image discussed above in Section VIII.A.2.k), which Smith also refers to as “the extracted signature value” (Smith, ¶ 39), qualifies as the claimed “stored signature” of the LLB code image. Madisetti, ¶ 108.

m) Limitation 1[e]

Smith discloses this limitation. Madisetti, ¶ 109. As explained above in Section VIII.A.2.k), the stored signature is created by applying the encryption key (i.e., UID 119). Madisetti, ¶ 109. That is, Smith recites that the LLB code image’s “signature value [is] signed over the code image according to [UID 119] through well-known hashing and encryption algorithms.” Smith, ¶¶ 39 (explaining such signature value of a code image), 40 (reflecting that the LLB code image is a code image to which paragraph 39’s discussion applies); Madisetti, ¶ 109.

n) Limitation 1[f]

Smith discloses this limitation. Madisetti, ¶ 110. The loaded LLB code image is executed if, and thus when, the determined current signature and the stored signature match. Madisetti, ¶ 110. Smith discloses comparing the determined current signature (“the derived signature value”) to the stored signature value (“the extracted signature value”) “to verify whether the code image is trusted.” Smith, ¶ 39; Madisetti, ¶ 110. Smith then explains that verification is successful “if the derived signature value and the extracted signature match with each other[, i.e., are identical].” Smith, ¶ 39; Madisetti, ¶ 110. Finally, Smith explains that the LLB code image is executed if the verification is successful, which a POSITA would have understood as synonymous with when the verification is successful. Smith, ¶ 40, FIG. 3 (showing blocks 307 and 311); Madisetti, ¶ 110. A POSITA would have

understood such execution as the execution of boot instructions contained within the LLB code image. Madisetti, ¶ 110.

Thus, Smith discloses “executing boot instructions contained in said signed portion of boot data, when said current signature and said stored signature are identical,” as claimed (Madisetti, ¶ 111), rendering claim 1 obvious.

o) Claim 2

Claim 2 is obvious for the reasons set forth in Section VIII.A.2.i) that the Mapped Limitation 1[a] Instructions Execution initializes (at least obviously) RAM 111 and loads the LLB code image from storage component 109 into initialized RAM 111. *See* Madisetti, ¶ 112.

p) Claim 3

For this claim, which depends from claim 2, Petitioner maps the recited signed portion of boot data to the iBoot code image, rather than the LLB code image. Madisetti, ¶ 113. The analysis of Limitation 1[a] in Section VIII.A.2.i) applies with equal force to loading the iBoot code image into RAM 111 for at least the following reasons. Madisetti, ¶ 113.

Smith’s FIG. 3 and the associated description thereof concern a process for secure booting. *See* Smith, ¶¶ 38-41, FIG. 3; Madisetti, ¶ 114. The process includes executing instructions (such as codes 115) stored in secure ROM 113 to locate a code image within the device (i.e., the claimed “computer system,” as explained in

Section VIII.A.2.h)) that is being securely booted. Smith, ¶ 38, FIG. 3; Madisetti, ¶ 114. The code image in question can be stored (and therefore located) in storage component 109. Smith, ¶ 38 (explaining the code image can be a “storage component” of the device, such as “a flash memory or a mass storage device,” which a POSITA would have understood to refer to storage component 109, which “may be a flash memory[] ... or other mass storage” device (*id.*, ¶ 31)); *id.*, ¶ 38 (explaining “process 300 may be performed by system 100 of FIG. 1”), FIG. 1; Madisetti, ¶ 114. Once found, the code image is then loaded “into a memory” (Smith, ¶ 38), which a POSITA would have understood to be either RAM 111 or memory component 103—the only memories into which Smith discloses loading a code (from storage component 109) using system 100 (Smith, ¶ 31). Madisetti, ¶ 114. After the loaded code image is verified based on UID 119 (Smith, ¶ 39)—which, as applied to the iBoot code image, satisfies Limitations 1[c]-1[e] for the same reasons discussed above in Sections VIII.A.2.k)-VIII.A.2.m) with respect to the LLB code image—the loaded code image is executed (Smith, ¶ 40). Madisetti, ¶ 114. Smith discloses iBoot 227 as one such code image (Smith, ¶ 40), which Smith shows loaded into RAM 111 in FIG. 2 (*id.*, FIG. 2). Madisetti, ¶ 114.

Correspondingly, claim 2—just as it was obvious for the reasons set forth in Section VIII.A.2.i) that the Mapped Limitation 1[a] Instructions Execution initializes (at least obviously) RAM 111 and loads the LLB code image from storage

component 109 into initialized RAM 111 based on the claim 1 mapping of the LLB code image to the signed portion of boot data—would also have been obvious for those same reasons applied to the iBoot code image, as explained in the preceding paragraph. Madisetti, ¶ 115.

Next, executing boot instructions contained in the iBoot code image (Smith, ¶ 40) includes the “initializing” and “loading” of Limitation 3[a] for at least the following reasons. Madisetti, ¶ 116. Smith describes the execution of the loaded and verified code image as the performance of booting operations that include “device hardware initialization.” Smith, ¶ 40; Madisetti, ¶ 116. With the iBoot code image loaded into RAM 111 (as explained *supra*), memory component 103 would have been device hardware that need not have been initialized at that point, and Smith elsewhere describes iBoot as a code image that, when executed, locates and loads “into a memory component of the device” the Kernalcache code image (Smith, ¶ 52⁷), which is “boot” data (as explained above in Section VIII.A.2.k)) and thus satisfies “said boot data.” See ’037 Patent, 6:34-44, 9:9-17, 10:14-31 (description

⁷ While this paragraph concerns a recovery process, Smith is addressing the same iBoot code image Smith addresses elsewhere, so a POSITA would have understood the iBoot code image to, when executed as described in Smith paragraph 41, perform the same operations described in Smith paragraph 52. Madisetti, ¶ 116 n.5.

relevant to FIG. 1c); Madisetti, ¶ 116. As a result, a POSITA would have interpreted the referenced “device hardware initialization” to include the initialization of memory component 103. Madisetti, ¶ 116. At a minimum, it would have been obvious to a POSITA for the iBoot code image to include boot instructions that caused such locating and loading of the Kernelcache code image because (1) a code image could not have been stored in memory component 103 until memory component 103 had been initialized, (2) Smith does not otherwise describe the initialization of memory component 103, and (3) Smith discloses executing the iBoot code image from RAM 111 as a predicate to locating the Kernelcache code image and loading same into memory component 103. *See* Madisetti, ¶ 116.

Finally, although Smith paragraph 41 does not state that the execution of the loaded and verified iBoot code image verifies the Kernelcache code image, Smith does disclose that such verification—which a POSITA would have understood as the claimed “verifying an integrity of [the Kernelcache code image]”—takes place, both (1) as part of the booting process depicted in FIG. 3 in which, after execution of the iBoot code image, control passes from block 317 to block 319, for finding the Kernelcache code image, then to block 303, for loading the Kernelcache code image into memory, and then to block 305, for verifying that Kernelcache code image (Smith, FIG. 3, ¶¶ 40-41), and (2) during a recovery process in which Smith describes verifying the located and loaded-into-memory Kernelcache code image to

be a trusted code image “according to a process such as described at block 305 of FIG. 3” (*id.*, ¶ 52). Madisetti, ¶ 117. A POSITA would have found it logical and straightforward (and, therefore, obvious) for the execution of the iBoot code image to have performed such verification, considering that: (1) the iBoot code image would have been trusted at that point, and (2) Smith discloses the execution of one code image can be used to verify a second code image (*see* Smith, claims 1-3 (describing the execution of a first verified code image verifies a second code image); *see also id.*, ¶ 54 (describing the execution of the Kernelcache code image locate, load, and verify an operating system component during a recovery process)). *See* Madisetti, ¶ 117. A POSITA would have considered a modification to Smith’s iBoot code image that caused the iBoot code image to, when executed, perform such verifying of the Kernelcache code image to be the application of a Smith-disclosed verification approach (i.e., one verified code image being executed to verify a second code image, as described in Smith claims 1-3) to achieve a Smith-disclosed verification of the Kernelcache code image (explained *supra*), which is one of only three disclosed code images (LLB, iBoot, and Kernelcache), while yielding nothing unpredicted. Madisetti, ¶ 117. This is akin to the obvious-to-try rationale supporting obviousness (*KSR*, 450 U.S. at 421) and/or to the rationale that a combination of familiar elements according to known methods that yields nothing more than predictable results is obvious (*KSR*, 450 U.S. at 416), and is therefore also obvious.

q) Claim 5

Claim 5 recites that the method of claim 1 further comprises “locking said encryption key for external access prior to executing said boot instructions.” As explained in Section VIII.A.2.i), it would have been obvious to try a cold reboot of the computer system, which would have cycled the power thereto off and on. *See* Madisetti, ¶ 118. That obvious-to-try turning off of the power to the computer system would have locked UID 119 for external access prior to executing the boot instructions (such executing having been addressed in Section VIII.A.2.n)), and been followed by the performance of claim 1’s limitations, as described above in Sections VIII.A.2.i)-VIII.A.2.n). *See* Madisetti, ¶ 118. Thus, Smith renders claim 5 obvious.

r) Claim 6

Smith discloses “determining whether an upgrade version of said boot data is available” because Smith discloses determining whether an updated version of the LLB code image (which qualifies as the claimed “boot data” as explained in Section VIII.A.2.k)) is available. Madisetti, ¶ 119. Smith discloses updating the software and the firmware of the computer system (*see* Smith, ¶ 6 (explaining the need for security and the flexibility to update the device’s software and firmware); Sec. VIII.A.2.h) (explaining why the claimed method concerns a computer system with reference to Section VIII.A.2.d)); Madisetti, ¶ 119. Smith also discloses a particular mode for accomplishing such updates: DFU mode, which stands for “device

firmware upgrade” mode. Smith, ¶ 38; *see also id.*, ¶ 43 (describing some purposes of, and some conditions for entering, DFU mode); Madisetti, ¶ 119. A POSITA would have understood Smith’s reference to “firmware” in “D[evice] F[irmware] U[pgrade]” mode to encompass boot data, such as the LLB code image, because Smith associates DFU mode with the recovery of Smith’s code images when they cannot be verified. Smith, ¶¶ 48-54, FIG. 7; Madisetti, ¶ 119.

In particular, Smith discloses the computer system—after not successfully locating and verifying the LLB code image—being coupled to a host computer, publishing a status on the computer system’s recovery state, receiving from the host computer an LLB code image, and loading and verifying that the received LLB code image is trusted, after which that received LLB code image is executed. Smith, ¶¶ 48-50, FIG. 7; Madisetti, ¶ 120. A POSITA would have understood the LLB code image received from the host computer to be an upgrade relative to at least an existing LLB code image that could not be verified; a POSITA would have understood such an unverified image to possess a problem of some kind (such as damage or corruption) that a POSITA would not have understood the new image to possess. Madisetti, ¶ 120.

Furthermore, a POSITA would have understood that Smith discloses determining whether an upgrade version of the LLB code image is available in each of the following ways, any one of which is adequate. First, the entry of the computer

system into DFU mode (shown in FIG. 7 as “DFU 713”) establishes such a determination because the computer system would not have entered that upgrade mode if no upgrade were available. Smith, ¶ 48, FIG. 7; Madisetti, ¶ 121. Second, the individual who “coupled” the computer system to the host computer for receiving the upgraded LLB code image made the determination that such upgraded code image was available; otherwise, they would not have coupled the computer system to the host computer. Smith, ¶ 49; Madisetti, ¶ 121. Third, the computer system’s publication of its status to the host computer and subsequent receipt of the host-computer-transmitted LLB code image (as described in Smith ¶ 49) involved determining such LLB code image was available. Madisetti, ¶ 121.

To the extent PO disagrees that Smith’s paragraphs 48-50 and FIG. 7 disclose the claimed determining, such determining is also disclosed in Smith paragraphs 59-60 and FIGs. 5 and 9. Madisetti, ¶ 122. There, Smith discloses establishing a network connection between the computer system and a host computer (such as is shown in FIG. 5), publishing by the computer system to the host computer a status indicating the computer system is an “update mode,” and receiving by the computer system from the host computer a code image that includes a software package related to an updated version of an application of the computer system. Smith, ¶¶ 59-60, FIGs. 9 and 5; Madisetti, ¶ 122. A POSITA would have understood the updated LLB code image received from the host computer to be an upgrade relative to the

non-updated LLB code image being replaced. Madisetti, ¶ 122. Moreover, because Smith explains that process 900 described in paragraphs 59-60 is performable with system 100 (Smith, ¶ 59), and the LLB code image is one of only three code images specifically identified by Smith for system 100 (*see* Smith, *e.g.*, ¶ 40), a POSITA would have understood the code image referenced in Smith paragraph 60 to apply to the LLB code image. Madisetti, ¶ 122.

Furthermore, a POSITA would have understood that Smith's paragraphs 59-60 and FIGs. 5 and 9 disclose determining whether an upgrade version of the LLB code image is available in each of the following ways, any one of which is adequate. Madisetti, ¶ 123. First, the entry of the computer system into "update mode" (*see* block 903 in FIG. 9) establishes such a determination because the computer system would not have entered that update mode if no update were available. Smith, ¶ 59, FIG. 9; Madisetti, ¶ 123. Second, the individual who enabled the establishment of "a network connection" between the computer system and the host computer for receiving the updated LLB code image made the determination that such updated code image was available; otherwise, they would not have enabled that connection between the computer system and the host computer. Smith, ¶ 59, FIGs. 5, 9; Madisetti, ¶ 123. Third, the computer system's publication of its status to the host computer and subsequent receipt of the host-computer-transmitted LLB code image

(as described in Smith ¶¶ 59-60 and shown in FIG. 9) involved determining such LLB code image was available. Madisetti, ¶ 123.

B. Ground 2 – Claim 20 Is Obvious Over Smith in View of Bolotin

Claim 20 is obvious over Smith in view of Bolotin.

1. Overview of Bolotin

Bolotin, which is entitled “Memory Lock System,” is analogous art to the invention of claim 20 because, as the balance of this Bolotin overview and the subsequent discussion of the motivation-to-combine reflects, Bolotin concerns at least one problem faced by the ’037 Patent’s inventors: controlling access to memory containing sensitive information. *See* Bolotin, Title; ’037 Patent, 3:11-25 (discussing controlling access to pre-boot information), 3:56-61 (discussing storing same in non-volatile internal memory); Madisetti, ¶ 126. In particular, Bolotin discloses an embodiment of a self-contained and independent memory lock system that “includes a controller, such as a micro-controller 212, connected by a communication channel 214 to a memory 216, such as a flash memory, micro disk, or other file storage system.” Bolotin, 5,⁸ 2, FIG. 2; Madisetti, ¶ 94. A POSITA would have understood such memory to be non-volatile. *See* ’037 Patent, 7:27-32 (indicating “flash memory” is a form of “non-volatile memory”); Madisetti, ¶ 126.

⁸ Bolotin citations are to native page numbers.

The microcontroller is also connected to a power connector 204. Bolotin, 5, FIG. 2; Madisetti, ¶ 126.

“The micro-controller 212 can have part of its operations dedicated to a data transfer function between the connector 204 and the memory 216 and another part to the locking function.” Bolotin, 5, FIG. 2; Madisetti, ¶ 127. “[P]ower for the memory lock system 200 can be drawn from the electronic connector 204 from the electronic system to which it is connected.” Bolotin, 5, FIG. 2; Madisetti, ¶ 127.

According to Bolotin:

Where the power for the memory lock system 200 is drawn from the electronic connector 204, the micro-controller 212 can use the locking function controller 215 to shut down the power to or reset the state of the data transfer function controller 213 to prevent or lock data transfer in the micro-controller 212. The micro-controller 212 can also shut down power to or otherwise disable the memory 216 to prevent or lock data transfer in the memory.”

Bolotin, 5, FIG. 2; Madisetti, ¶ 127.

Bolotin explains that “compare circuitry or firmware” of memory lock system 200, which a POSITA would have understood to correspond to the schematically-illustrated micro-controller 212 and locking function controller 215 of FIG. 2, is what renders the system unlocked and operable. Bolotin, 5, 8-9; Madisetti, ¶ 128. A POSITA would have therefore understood from this that Bolotin discloses

circuitry for locking and enabling power to reach a memory containing sensitive information. Madisetti, ¶ 128.

2. Motivation to modify Smith's chip 105 chip to include circuitry like that provided by Bolotin's micro-controller 212 and locking function controller 215

A POSITA would have been motivated to modify Smith's chip 105 with circuitry like that provided by Bolotin's micro-controller 212 and locking function controller 215 to provide a self-contained means in Smith's computer system for enabling or preventing external access to sensitive information stored in ROM 113 (such as UID 119). Madisetti, ¶ 129. As a POSITA would have known, the proliferation of computing devices resulted in an acute need to improve the security of computer systems, including hand-held computing devices. *See* Bolotin, 1 ("Security is a critical issue with almost all aspects of computer use and mobile electronic device use."), 2-3 (security for the data on portable storage devices and electronic products using them "is greater than ever"); Madisetti, ¶ 129.

Bolotin teaches circuitry (Bolotin, 5, 8-9, FIG. 2) that can shut down or enable power to a memory in order to lock or enable data transfer with the memory. Madisetti, ¶ 130. As a POSITA would have appreciated from Bolotin's disclosure, controlling power to a memory through circuitry was a straightforward and effective way to predictably and securely control access to the memory. Bolotin, 5, 8-9, FIG. 2; Madisetti, ¶ 130. Bolotin also teaches that its memory lock systems (which

includes system 200) can be integrated internally with computer systems like “iPODs.” Bolotin, 8; Madisetti, ¶ 130. Bolotin thus teaches the desirability of using circuitry to allow or shut off power to memory as an effective means of allowing or preventing access to the memory in devices like the computer system disclosed in Smith. Bolotin, 5, 8-9, FIG. 2; Sec. VIII.A.2.d) (explaining why Smith discloses secure booting of a computer system like an Apple iPod®); Madisetti, ¶ 130.

3. Smith’s chip 105, modified in view of Bolotin, meets claim 20

Motivated as explained above, a POSITA would have modified Smith’s chip 105 in view of Bolotin to provide circuitry like that provided by Bolotin’s micro-controller 212 and locking function controller 215 for achieving Smith’s already-disclosed functions of enabling or disabling access to at least the portion of secure ROM 113 that includes UID 119 (*see* Sec. VIII.A.2.c)). Madisetti, ¶ 131. As modified, Smith’s chip 105 would have included circuitry for enabling/disabling access to ROM 113 (and, thus, UID 119) under the conditions discussed in Section VIII.A.2.c). Madisetti, ¶ 131. Because Smith’s ROM 113 is on chip 105 along with chip 105’s CPU core (*see* Section VIII.A.2.a) and its subsections), it would have been straightforward for a POSITA to have modified chip 105 to predictably implement such circuitry in view of the above-discussed disclosures of Bolotin. Madisetti, ¶ 131. Such circuitry meets the corresponding-structure circuitry under a

means-plus-function construction of the “locking mechanism” element (*see* Sec. VII.D), *see* Madisetti, ¶ 131, thereby rendering claim 20 obvious.

C. Grounds 3 and 4 – Claims 19 and 21 Are Obvious Over Smith in View of Little (Ground 3), and Claim 20 Is Obvious Over Smith in View of Little and Bolotin (Ground 4)

Grounds 3 and 4 are the same as Grounds 1 and 2, respectively, except for the addition of Little with respect to Limitations 19[pre]-19[a] (Section VIII.A.2.a)) and 21[a]-21[b][i] (Sections VIII.A.2.e)-VIII.A.2.f)). *See* Madisetti, ¶ 135.

As explained in VIII.A.2.a) (and, in particular, Section VIII.A.2.a)(3)), Smith’s system includes a CPU core that implements his secure-boot processes, and it would have been obvious for Smith’s chip 105 to include it. To the extent that PO contends otherwise, Little discloses an “on-chip ROM [that] is physically integrated with [a] processor” (Little, Abstract; *see also id.*, ¶¶ 7, 30, 35, FIG. 2), and it would have been obvious to configure Smith’s chip such that Smith’s CPU core was accordingly integrated with Smith’s on-chip ROM 113. *See* Madisetti, ¶ 132.

Little’s ROM 19 “contains the authentication code, including one or more public keys, required for authenticating the program software residing in external memory 116 in known fashion.” Little, ¶ 31, FIG. 2; Madisetti, ¶ 133. Smith’s ROM 113 also stores “a fingerprint 123 based on a signature from a trusted entity such as a public key certificate” that can be used to verify whether an operating system component stored in storage component 109 can be trusted and therefore

executed. Smith, ¶¶ 32, 35, FIG. 2; Madisetti, ¶ 133. And like Little’s authentication code in Little’s ROM 19, Smith teaches that the storage of fingerprint 123 in Smith’s ROM 113 is secure. *See* Little, Abstract (touting that “the code stored in ROM can neither be tampered with nor replaced, and can therefore be assumed at all times to be authentic”), ¶¶ 5, 7, FIG. 2; Smith, ¶¶ 29 (the “secure ROM” stores the fingerprint), 28 (describing the secure ROM and that its content may be stored during the manufacturing stage), FIGs. 1 and 2 (depicting “Secure ROM” 113 containing “Fingerprints” 123); Madisetti, ¶ 133.

To provide such security, Little explains that “[t]he on-chip ROM is physically integrated with the processor itself, so the code stored in ROM can neither be tampered with or replaced.” *See* Little, Abstract; *see also id.*, ¶¶ 7, 34; Madisetti, ¶ 134. Particularly in view of their aligned goals (*supra*), it would have been obvious to do the same in Smith—i.e., incorporating Smith’s “Secure ROM” (i.e., ROM 113) into Smith’s CPU core—to achieve the same benefit. *See* Madisetti, ¶ 134.

D. Ground 5 – Claims 5-6 Are Obvious Over Smith in view of Challenger

Claims 5-6, which depend from claim 1, are obvious over Smith in view of Challenger. *See* Madisetti, ¶ 136.

1. Overview of Challenger

Challenger is analogous art to the ’037 Patent at least because both concern the secure storage of sensitive information in computer systems. *See* ’037 Patent, 3:11-

19; Challenger, ¶ 1; Madisetti, ¶ 137. In addition, and further reflecting that Challenger is analogous art to the '037 Patent, both concern ensuring a symmetric encryption key is securely stored in a computer system. *See* '037 Patent, Abstract; Challenger, Abstract; Madisetti, ¶ 137.

Challenger discloses making available (i.e., unlocking), at power-on, system configuration and access data—that, during POST (Power-On-Self-Test), is “used to verify the identity [of the] user and to determine the appropriate access rights and Boot devices”—in a non-volatile memory. Challenger, ¶ 2; Madisetti, ¶ 138. Challenger also discloses locking the system configuration and access data from external access after such use of same but before executing boot instructions. Challenger, ¶ 2; Madisetti, ¶ 138. Specifically, Challenger recites: “Just before boot, POST ... locks the system configuration and access data in a [non-volatile random access memory] NVRAM device.” Challenger, ¶¶ 2, 18; Madisetti, ¶ 138. As a POSITA would have appreciated, the aspect or aspects of the system configuration and access data Challenger locks just before boot were not needed during boot and experienced improved security as a result of the locking. Challenger, ¶ 2; Madisetti, ¶ 138.

2. Motivation to Combine

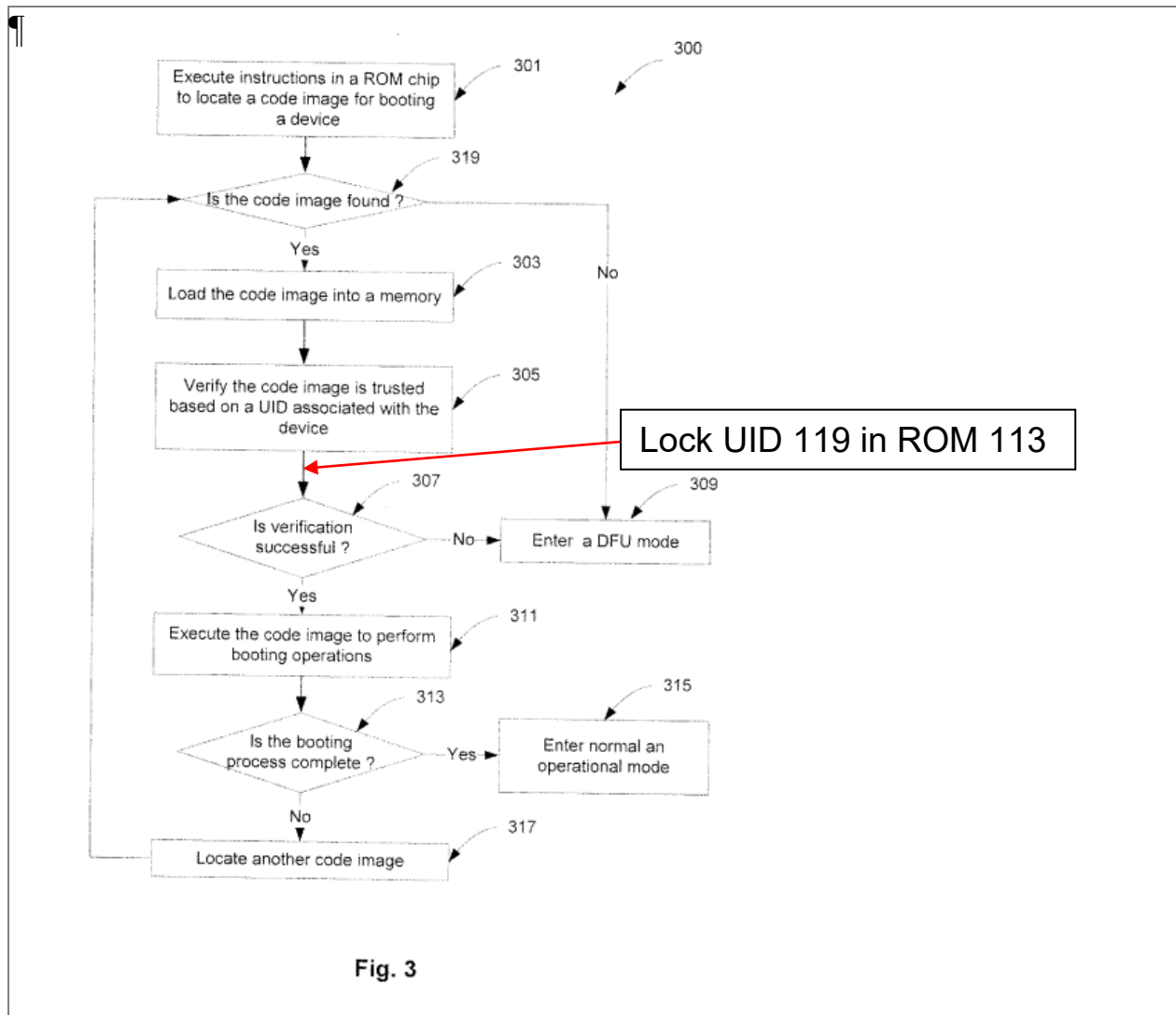
A POSITA would have been motivated to modify Smith’s system 100 (including the relevant processing logic of process 300 performable by system 100)

in view of Challener to lock external access to UID 119 (in ROM 113) after UID 119 is used to verify the LLB code image is trusted but before executing the LLB code image (which Smith characterizes as the performance of booting operations (Smith, FIG. 3 (block 311), ¶ 40), similar to Challener's disclosure of using system configuration and access data to verify a user's identity and determine appropriate access rights and then locking such data prior to booting (Challener, ¶ 2). Madisetti, ¶ 139. Furthermore, a POSITA also would have been motivated to modify the processing logic of process 300 to unlock UID 119 just prior to the time it is next needed, which would have been to verify the next code image (as reflected in the process flow of Smith's FIG. 3) to enable the use of UID 119 as Smith describes. *See* Smith, ¶ 41, FIG. 3 (showing that, when the booting process is not complete, the process flow includes the verification of another code image using UID 119); Madisetti, ¶ 139. As a POSITA would have appreciated, in both Challener and Smith, the locked data is not (or would not be) needed during the locked time period, and external access to the locked data might have otherwise been possible. Madisetti, ¶ 139.

3. Modification of Smith in view of Challener

Motivated as discussed above, a POSITA would have found it logical and straightforward (and, therefore, obvious) to modify Smith's system 100 (including the relevant processing logic of process 300 performable by system 100) to lock UID

119 (in ROM 113) to external access just after UID 119 was used to verify the LLB code image (and, thus, just before the verified LLB code image was executed) and then unlock UID 119 just prior to its use to verify the next code image. Madisetti, ¶ 140. A POSITA had the knowledge and skill to accomplish this by modifying the processing logic of process 300. *See* Smith, ¶¶ 38-41 (describing how process 300 is performed by system 100 using process logic), FIG. 3; Madisetti, ¶ 140. Specifically, a POSITA would have modified such processing logic to lock UID 119 to external access after block 305 (where it is used to verify the LLB code image) and prior to block 311 (where the LLB code image is executed to perform booting operations), and then unlock UID 119—as necessary—prior to verifying the next code image (located and found at blocks 317 and 319, respectively) at the next performance of block 305 following a determination that the booting process is not complete (at block 313). Madisetti, ¶ 140. This obvious locking modification is shown in the annotated version of Smith FIG. 3 below (the unlocking modification has not been shown):



Madisetti, ¶ 140.

Smith confirms the feasibility of these modifications. Madisetti, ¶ 141. Specifically, Smith discloses that, in another process involving the execution of an unverified code image, processing logic can be used to “disable access[] to a UID of a secure ROM in a device such as UID 119 [stored in ROM 113] in FIG. 1.” Smith, ¶ 61; Madisetti, ¶ 141. Smith discloses additional techniques for disabling access to UID 119 depending on computer system diagnostic or testing requirements

(configuring a trusted code image to, when executed, “turn off access[] to the UID” and using a hardware switch with settings “that turn off access[] to the UID”) that: further reinforce that a POSITA had the skill to perform the proposed modification, and reflect that controlling external access to UID 119 under certain circumstances was warranted. Smith, ¶ 61; Madisetti, ¶ 141.

4. Element-by-Element Analysis

a) Claim 5

Smith, modified as discussed above in Section VIII.D.3 in view of Challenger, includes locking UID 119 (in ROM 113) for external access prior to the execution of the LLB code image. Madisetti, ¶ 142. Such locking is depicted in the annotated version of Smith FIG. 3 depicted above, which occurs after UID 119 is used as described in Section VIII.A.2.k) to determine a current signature of the loaded LLB code image (corresponding to block 305 depicted in FIG. 3 and described in Smith paragraph 39) and before execution of the loaded LLB code image (corresponding to block 311 depicted in FIG. 3 and described in Smith paragraph 40). Smith, FIG. 3, ¶¶ 39-40; Madisetti, ¶ 142.

b) Claim 6

Smith discloses the “determining” of claim 6, which depends from claim 5, for the same reasons given above in Section VIII.A.2.r). Madisetti, ¶ 143. Claim 6 is therefore obvious over Smith in view of Challenger.

IX. DISCRETIONARY DENIAL UNDER SECTION 325(d) IS IMPROPER

A. Prosecution History

In a first office action, the examiner rejected claims 1-4, 19 and 21 as anticipated by Little. '037 PH, 138-142. Little discloses a method of booting a computer system using a public key stored in internal ROM. Little, Abstract, ¶¶ 30-35. The examiner also rejected claims 5 and 20 as obvious over little in view of Chen. '037 PH, 142-143. Chen teaches a similar booting method in which the encryption key is locked from external access before executing boot instructions. *Id.*

The Applicants responded by arguing that Little is silent with respect to the stored signature being created by applying the encryption key, and Little only describes using public keys for authenticating a digital signature, and does not describe using encryption keys for creating a digital signature. *Id.*, 172-173.

In a second office action, the examiner rejected claims 1-4, 19 and 21 as anticipated by Bhatt. *Id.*, 182-184. Bhatt discloses a method of booting a computer system using a verification code stored in boot storage to generate a signature used to compare to a signature value stored in ID storage. *Id.*, 182-183. Boot code is executed if the computed signature and the signature value match. *Id.*, 182-183.

The Applicants again responded without amendments and argued that while Bhatt discloses that a signature is computed using a hashing or similar algorithm using a secret key, Bhatt does not disclose that the key is stored in internal memory,

and that it is that secret key that corresponds to the encryption key of the claims. *Id.*, 205-206.

In a third office action, the examiner rejected claims 1-5 and 19-21 as being anticipated by Bhatt in view of Kumar. *Id.*, 304-306. Kumar discloses a method involving secure booting, including use of an internal non-volatile memory area containing an encryption key. *Id.*, 305.

Once again, the Applicants responded without amendment and argued that while Kumar describes storing a cryptographic key called a PUK in a non-volatile form, Kumar teaches that the key is used to encrypt and protect another key, and therefore the PUK is not an encryption key used to determine a current signature of a portion of boot data. *Id.*, 326-327. Applicants also argued that Bhatt teaches away from the claims and from storing previously calculated signatures with portions of boot code in external memory. *Id.*, 328. The examiner then allowed the claims without further remarks. *Id.*, 333-337.

B. The Grounds Are Not Based on Substantially the Same Prior Art or Arguments Previously Considered by the Office

The proposed Grounds are not based on substantially the same prior art or arguments the Office has already considered. Previously-unconsidered Smith is more material than the considered art at least because Smith discloses a symmetric encryption key stored in non-volatile memory that is used to verify boot code by comparing a current signature computed using the symmetric encryption to a stored

signature of the boot code generated using the same encryption key. *See e.g.*, Section VIII.A.2.b)(2) (addressing Limitation 19[b]). Smith's disclosure also adheres more closely to the '037 Patent's other independent claim limitations and its challenged dependent claims than any other considered reference. *See e.g.*, Section VIII.A.2. Moreover, none of the arguments regarding Smith, nor Dr. Madisetti's supporting testimony, were previously presented to the Office. For at least these reasons, denial of institution under Section 325(d) would be inappropriate.

X. CONCLUSION

Institution is appropriate.

Dated: April 20, 2022

Respectfully submitted,

/Mark Garrett/
Mark Garrett

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(a)(1)(i), the undersigned certifies that this Paper—exclusive of the table of contents, table of authorities, mandatory notices under § 42.8, certificate of service, this certificate of word count, appendix of exhibits, and claim listing—includes 13,961 words based on the word count of the word-processing system used to prepare the paper.

/Mark Garrett/
Mark Garrett (Reg. No. 44,699)

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. § 42.6(e) and 37 C.F.R. § 42.105(a), the undersigned certifies that on April 20 2022, a complete copy of this Petition for *Inter Partes* Review, Petitioner's power of attorney, and all exhibits were served on Patent Owner at the correspondence addresses of record listed below by FedEx®:

McClure, Qualey & Rodack, LLP
280 Interstate North Circle SE
Suite 550
Atlanta GA 30339

/Mark Garrett/
Mark Garrett (Reg. No. 44,699)