UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NINTENDO CO., and NINTENDO OF AMERICA INC.,
Petitioners

v.

ANCORA TECHNOLOGIES, INC.,
Patent Owner

Case IPR2021-01338
Patent No. 6,411,941

DECLARATION OF DR. DAVID MARTIN

TABLE OF CONTENTS

List of Exhibits

EXHIBIT NO.	TITLE
1001	U.S. Patent No. 6,411,941 to Mullor et al. ("'941 Patent")
1002	Image File Wrapper of U.S. Patent No. 6,411,941 ("File History")
1003	Declaration of Andrew Wolfe, Ph.D. ("Wolfe Decl.")
1004	U.S. Patent No. 4,658,093 ("Hellman")
1005	U.S. Patent No. 5,892,906 ("Chou")
1006	U.S. Patent No. 5,933,498 ("Schneck")
1007	Reserved
1008	Reserved
1009	Reserved
1010	Reserved
1011	Claim Construction Order, *Ancora Techs., Inc. v. Apple Inc.*, No. 4:11-cv-06357 (N.D. Cal. Dec. 31, 2012) (ECF No. 107).
1012	Final Claim Constructions of the Court, *Ancora Techs., Inc. v. LG Elecs., Inc.*, No. 1:20-cv-00034 (W.D. Tex. June 2, 2020) (ECF No. 69).
1013	Supplemental Claim Construction Order, *Ancora Techs., Inc. v. LG Elecs., Inc.*, No. 1:20-cv-00034 (W.D. Tex. Aug. 19, 2020) (ECF No. 93).
1014	Civil Minutes re Telephonic Markman Hearing, *Ancora Techs., Inc. v. TCT Mobile (US), Inc.*, No. 8:19-cv-02192 (C.D. Cal. Nov. 12, 2020) (ECF No. 66) (attaching "The Court's Final Ruling on Claim Construction (*Markman*) Hearing," but also ordering further meet and confer on subject).
1015	Civil Minutes re Telephonic Markman Hearing, *Ancora Techs., Inc. v. TCT Mobile (US), Inc.*, No. 8:19-cv-02192 (C.D. Cal. Nov. 19, 2020) (ECF No. 69) (confirming no change to "The Court's Final Ruling on Claim Construction (*Markman*) Hearing").
1016	Decision Granting Institution of Inter Partes Review, TCT Mobile (US) Inc. v. *Ancora Technologies, Inc.*, No. IPR2020-01609 (Feb. 16, 2021) (Paper No. 7) ("TCL Institution Decision").
1017	Decision Granting Institution of Inter Partes Review, *Sony Mobile Commc'ns AB v. Ancora Technologies, Inc.*, No.

	IPR2021-00663 (June 10, 2021) (Paper No. 17) ("Sony Institution Decision").
2001	Microsoft Corporation's Request for *Ex Parte* Reexamination Image File Wrapper, Control No. 90010560
2002	*Ancora Techs., Inc. v. Apple, Inc.*, 744 F.3d 732 (Fed. Cir. 2014)
2003	Deposition Excerpts of Jon Weissman, *Ancora Techs., Inc. v. HTC America, Inc.*, Case No. 2:16-cv-01919 (W.D. Wa. Sep. 9, 2019)
2004	Declaration of Ian Jestice, *Ancora Techs., Inc. v. HTC America, Inc.*, Case No. 2:16-cv-01919 (W.D. Wa. Aug. 26, 2019)
2005	Brief of Appellees HTC America, Inc. and HTC Corporation, *Ancora Techs., Inc. v. HTC America, Inc., HTC Corp.*, Case No. 18-1404 (Fed. Cir. Apr. 23, 2018)
2006	Declaration of Jon Weissman, *Ancora Techns., Inc. v. HTC America, Inc.*, Case No. 2:16-cv-01919 (W.D. Wa. Sep. 4, 2019)
2007	Terplan, Kornel, Morreale, Patricia, THE TELECOMMUNICATIONS HANDBOOK, CRC Press, 2000
2008	COMPUTER USER'S DICTIONARY, Microsoft Press, 1998
2009	MICROSOFT COMPUTER DICTIONARY FOURTH EDITION, Microsoft Press,1999
2010	PC MAGAZINE ENCYCLOPEDIA, available at https://www.pcmag.com/encyclopedia (excerpt, definition of "Agent")
2011	U.S. Patent No. 6,411,941 File History with Beeble White Paper
2012	Joint Claim Construction Chart, *Ancora Techs., Inc. v. TCT Mobile (US) Inc., Huizhou TCL Mobile Commc'n Co., Ltd., and Shenzhen TCL Creative Cloud Tech. Co., Ltd.*, Case No. 8:19-cv-02192 (Dkt. #49, 49-1, 49-2)
2013	Declaration of Dr. David Martin, Ph.D., *Sony Mobile Commc'ns AB, Sony Mobile Commc'ns, Inc., Sony Elecs. Inc., and Sony Corp. v. Ancora Techs., Inc.*, IPR2021-00663, Ex. 2015 (PTAB Apr. 23, 2021)
2014	U.S. Patent No. 6,189,146 (Misra)
2015	U.S. Patent No. 5,479,639 (Ewertz)
2016	Decision Denying Institution of Inter Partes Review, *Samsung Electronics Co., Ltd. and Samsung Electronics America, Inc. v. Ancora Technologies, Inc.*, IPR2020-01184

2017	Declaration of Ian Jestice, *Ancora Technologies Inc. v. LG Electronics Inc., LG Electronics U.S.A. Inc., Samsung Electronics Co., Ltd., and Samsung Electronics America, Inc.*, Case No. 1:20-cv-00034 (Dkt. # 44-8)
2018	Declaration of Dr. David Martin (May 3, 2022)
2019	U.S. Patent No. 5,892,900 (Ginter)
2020	U.S. Patent No. 5,734,819 (Lewis)
2021	U.S. Patent No. 6,153,835 (Schwartz)
2022	*Ancora Techs. Inc. v. Apple, inc.*, Case No. 4:11-cv-06357 (Dkt. # 171-3) [Apple Inc.'s N.D. Cal. L.R. 3-3 (Invalidity) Disclosures]
2023	Petition, *HTC Corp. v. Ancora Techs. Inc.*, Case No. CBM2017-00054, Paper 1 (PTAB May 26, 2017)
2024	Institution Decision, *HTC Corp. v. Ancora Techs. Inc.*, Case No. CBM2017-00054, Paper 7 (PTAB Dec. 1, 2017)
2025	Croucher, "The BIOS Companion" (1997) (Excerpts)
2026	Transcript of Deposition of Dr. Andrew Wolfe (April 22, 2022)
2042	Barron's Dictionary of Computing and Internet Terms (5th Ed. 1996)
2043	Decision Granting Institution, Case No. IPR2020-01609, Paper 7 (PTAB Feb. 16, 2021)
2044	Free On-Line Dictionary of Computing (June 6, 1999), available at http://foldoc.org/bios
2045	U.S. Patent No. 5,684,951 (Goldman)
2046	Paul C. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, Proceedings of 16th Annual Int'l Cryptology Conf. (Aug. 18–22, 1996)

I, David Martin, hereby declare as follows:

1. I am making this declaration at the request of Patent Owner, Ancora Technologies, Inc., to investigate and opine on certain issues relating to the *Inter Partes* review of U.S. Patent No. 6,411,941 ("the '941 patent"). The Petition requests that the Patent Trial and Appeal Board ("PTAB" or "Board") review and cancel claims 1–3, 6–14, and 16 of the '941 patent.

2. I am being compensated for my work in this matter at a rate of $525 per hour. My compensation in no way depends on the outcome of this proceeding.

3. In preparation of this declaration, I have studied the exhibits as listed in the Exhibit List shown above.

4. In forming the opinions expressed below, I have considered:

   a. The documents listed above as well as additional patents and documents referenced herein;

   b. The relevant legal standards, as described further below, and any additional documents cited in the body of this declaration; and

   c. My knowledge and experience based upon my work and study in this area as described below.

I. Qualifications and Professional Experience

5. I have provided my full background in the curriculum vitae that is attached to this Declaration as Appendix A.

6. In this report, I respond to the expert opinions provided by Dr. Andrew Wolfe in the declaration ("Wolfe Declaration" or "Ex. 1003") submitted in IPR2021-01338 and filed by Nintendo Co., and Nintendo of America Inc. (hereinafter "Nintendo" or "Petitioners"). I am submitting a similar analysis of Dr. Wolfe's substantively identical declaration in IPR2021-01406 filed by Roku, Inc. and Vizio, Inc. (hereinafter "Roku/Vizio"). I responded to a similar declaration submitted by Dr. Wolfe and filed as Ex. 1003 in IPR2020-01609, in the context of my previous work in IPR2021-00663.

7. This declaration is based on my study of the material that was available to me at the time of its writing. I reserve the right to update, supplement, or amend this rebuttal expert report in view of additional information obtained through discovery or other information that might become available between now and trial that is significant to the opinions set forth in herein.

8. In this declaration, due to my understanding of the law, I am not offering an opinion regarding each cited reference or combination of references beyond responding to the opinions and evidence identified in the Wolfe Declaration. Further, I have not responded to each and every assertion made in the Wolfe Declaration and instead have focused on what I consider to be the clearest faults in its arguments and proofs. My decision not to address an issue or argument thus should not be understood as a tacit admission that I agree with or do not dispute the

positions of Dr. Wolfe or of the Nintendo or Roku/Vizio petitioners.

9. I further note that, except where stated otherwise, I have assumed for purposes of my analysis that each reference or combination cited in the Wolfe Declaration actually constitutes prior art.

II. Background of this Matter

10. It is my understanding that Petitioners filed a Petition in IPR2021-01338, requesting an *inter partes* review ("IPR") of claims 1−3, 6−14, and 16 ("the Challenged Claims") of U.S. Patent No. 6,411,941. It is my understanding that Ancora filed a preliminary response to the Petition, which cited my declaration from IPR2021-00663. It is further my understanding the Patent Trial and Appeal Board instituted trial based on the Petition on January 27, 2022.

III. Summary of Opinions Regarding the Validity of the '941 Patent

11. In the Wolfe Declaration, Dr. Wolfe opined that claims 1–2, 11, and 13 of the '941 Patent are invalid as obvious based on U.S. Patent No. 4,658,093 (Ex. 1004, hereinafter "Hellman") in view of U.S. Patent No. 5,892,906 (Ex. 1005, hereinafter "Chou"), and that claims 1–3, 6–14, and 16 of the '941 Patent are invalid as obvious based on Hellman in view of Chou and U.S. Patent No. 5,933,498 (Ex. 1006, hereinafter "Schneck"). (Ex. 1003 at ¶ 27.)

12. I disagree with many of Dr. Wolfe's opinions. In this declaration, I

provide my opinions in response to the various allegations of invalidity made in the Wolfe Declaration, as well as the bases for my opinions. Based on my review of the ’941 patent, the Wolfe Declaration, and the references addressed therein, it is my opinion that the references cited by Dr. Wolfe do not disclose every element of the Challenged Claims.

13. All of the opinions contained within this declaration are based on my own personal knowledge and professional judgment. If called as a witness in this matter, I am prepared to testify about these opinions.

IV. High-Level Description of Materials Studied

14. Before writing this declaration, I studied the ’941 patent and its file history. I also studied the Wolfe Declaration and the references cited therein. I further considered additional material noted in this report.

15. I have also reviewed claim constructions applied in related IPRs and district court lawsuits. (Ex. 1011, Ex. 1012, Ex. 1013, Ex. 1014, Ex. 1015, Ex. 2043.)

16. For all terms found in the Challenged Claims, I have used the plain and ordinary meaning of the term as it would be understood by a person of ordinary skill in the art as of the time of the invention of the ’941 patent, as discussed below.

17. I conclude that the Challenged Claims of the ’941 patent are not invalid in view of the asserted Hellman and Chou references, in the combination asserted by Dr. Wolfe, as discussed further below.

V. Relevant Legal Principles

18. Although I am not an attorney, I have been advised by the attorneys for Ancora of certain legal principles as they relate to forming opinions as to the issues of validity of the Asserted Claims. I have applied this law to the facts set forth in this report in rendering my opinions. This section of my expert report provides my understanding of the legal principles that I have used in formulating my opinions.

A. Burden of Proof

19. It is my understanding that the petitioner in an *inter partes* review proceeding has the burden of proving a proposition of unpatentability by a preponderance of the evidence. I understand this to require the petitioner to show that there is a greater than 50% chance that the challenged claims are unpatentable.

B. Anticipation

20. I understand that "anticipation" under 35 U.S.C. § 102 exists only if a single prior art reference or product discloses or contains, expressly or inherently, each and every limitation of the claim at issue. In other words, every limitation of the claim must identically appear in a single prior art reference for the reference to anticipate that claim.

21. I also understand that all elements of the claim must be disclosed in the reference as they are arranged in the claim. I also understand that, to be considered anticipatory, the prior art reference must be enabling and must describe the

patentee's claimed invention sufficiently to have placed it in the possession of a person of ordinary skill in the field of invention. I further understand that the relevant standards for what constitutes "prior art," for purposes of anticipation under the relevant paragraphs of §102, are as follows (emphases added):

> (a) [T]he invention was ***known or used by others*** in this country, or patented or described in a ***printed publication*** in this or a foreign country, before the invention thereof by the applicant for patent, or
>
> (b) [T]he invention was patented or described in a ***printed publication*** in this or a foreign country or ***in public use or on sale*** in this country, more than one year prior to the date of the application for patent in the United States, or
>
> . . .
>
> (e) [T]he invention was described in—(1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) ***a patent granted on an application for patent by another filed in the United States*** before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for the purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language
>
> . . .

> (g)(2) [B]efore such person's invention thereof, the ***invention*** was made in this country by another inventor who had not abandoned, suppressed, or concealed it. In determining priority of invention under this subsection, there shall be considered not only the respective dates of conception and reduction to practice of the invention, but also the reasonable diligence of one who was first to conceive and last to reduce to practice, from a time prior to conception by the other.

22. To be "known or used by others in this country" under § 102(a), the knowledge or use of the invention must be accessible to the public. If a process is used in secret, and if the public is unable to learn the process by examining the product that is eventually sold, then that process is not publicly accessible.

23. I understand that a "public use" under 35 U.S.C. § 102(b) may be established by showing a public, non-secret, non-experimental use of the invention in the United States prior to the critical date. Use of an invention may be public where it is exposed or demonstrated to persons other than the inventor, who are under no obligation of secrecy and where there is no attempt to keep the device from the public.

24. I understand that, for purposes of § 102, the term "printed publication" means a publication that is sufficiently accessible to the public interested in the art. I understand that the critical factor for determining whether a reference constitutes a "printed publication" under § 102 is "public accessibility." I further understand that

a reference is "publicly accessible" only if Petitioners make "a satisfactory showing that such document has been disseminated or otherwise made available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, can locate it and recognize and comprehend therefrom the essentials of the claimed invention without the need of further research or experimentation." *Bruckelmyer v. Ground Heaters, Inc.*, 445 F.3d 1374, 1378 (Fed. Cir. 2006) (quoting *I.C.E. Corp. v. Armco Steel Corp.*, 250 F. Supp. 738, 743 (S.D.N.Y. 1966)).

C. Obviousness

25. I understand that, in order to be considered as "prior art" for purposes of the § 103 obviousness inquiry, a reference must first qualify as "prior art" under one of the definitions stated above in the context of § 102.

26. In order to be considered as "prior art" for purposes of the § 103 obviousness inquiry, a reference must also be "analogous" to the Patent-in-Suit. I understand that a reference is "analogous" if (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the reference is reasonably pertinent to the problem faced by the inventor (even if it is not in the same field of endeavor as the claimed invention).

27. I understand that the relevant standard for obviousness is as follows:

> A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

35 U.S.C. § 103(a) (pre-America Invents Act); Manual of Patent Examining Procedure § 2141.

28. Stated another way, to show that a patent is "obvious" based on an alleged prior art reference or a combination of such references, it must be shown that a person of ordinary skill in the art would have been motivated to combine the teachings of the prior art references to achieve the claimed invention, and that such person of ordinary skill in the art would have had a reasonable expectation of success in doing so. To do this, a Defendant must show how and why a skilled artisan would have had a reason to combine the interrelated teachings of the prior art references.

29. Petitioners may show that a claim is invalid for obviousness under 35 U.S.C. § 103 if they demonstrate that two or more prior art references in combination disclose, expressly or inherently, every claim limitation so as to render the claim, as a whole, obvious. Alternatively, Petitioners may show that a claim is invalid for obviousness under 35 U.S.C. § 103 if a single prior art reference combined with the

knowledge of one of ordinary skill in the art discloses every claim limitation so as to render the claim, as a whole, obvious.

30. It is my understanding that in assessing the obviousness of claimed subject matter, one should evaluate obviousness over the prior art from the perspective of one of ordinary skill in the art at the time that invention was made (and not from the perspective of either a layman or a genius in that art). The question of obviousness is to be determined based on:

(a) The scope and content of the prior art;

(b) The difference or differences between the subject matter of the claim and the prior art (whereby in assessing the possibility of obviousness one should consider the manner in which a patentee and/or a Court has construed the scope of a claim);

(c) The level of ordinary skill in the art at the time of the alleged invention of the subject matter of the claim; and,

(d) Any relevant objective factors (the "secondary considerations") indicating non-obviousness. It is also my understanding that the United States Supreme Court clarified the law of obviousness in *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398 (2007) ("*KSR*" herein). I have read that opinion and incorporate it here by reference. Based on *KSR*, it is my understanding that to determine whether it would have been obvious to combine known elements in a manner claimed in a patent, one may consider such things as the interrelated teachings of multiple patents, the effects of demands known to the design community or present in the marketplace, and the background knowledge of one with ordinary skill in the art. The secondary considerations at issue may include commercial success of a product using the invention, if that commercial success is due to the invention; long-felt need for the invention; evidence of copying of the claimed invention; industry acceptance; initial skepticism; failure of others; and praise of the invention.

31. I also understand that while an analysis of any teaching, suggestion, or motivation to combine elements from different prior art references is useful in an obviousness analysis, the overall inquiry must be expansive and flexible.

32. I further understand that it is impermissible to use a hindsight reconstruction of references to reach the claimed invention without any explanation as to how or why the references would be combined to produce the claimed invention.

33. Moreover, I understand that a patent composed of several elements is not proved obvious merely by demonstrating that each of its elements was, independently, known in the prior art. But multiple prior art references or elements may, in some circumstances, be combined to render a patent claim obvious. I understand that I should consider whether there is an "apparent reason" to combine the prior art references or elements in the way the Challenged Claim does. Requiring a reason for the prior art combination protects against distortion caused by hindsight. Along the same lines, one cannot use the Patent-in-Suit as a blueprint to piece

together the prior art in order to combine the right ones in the right way as to create the claimed invention(s).

34. To determine whether there is such an "apparent reason" to combine the prior art references or elements in the way that a Challenged Claim does, it may be necessary to look to the interrelated teaching of multiple references, to the "effects of demands known to the design community or present in the marketplace, and to the background knowledge possessed by a person having ordinary skill in the art." *KSR* at 418.

35. I also understand that when the prior art teaches away from combining prior art references or certain known elements, discovery of a successful means of combining them is more likely to be non-obvious. A prior art reference may be said to teach away from a claim when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the claim or would be led in a direction divergent from the path that was taken by the claim. Additionally, a prior art reference may teach away from a claimed invention when substituting an element within that prior art reference for a claim element would render the claimed invention inoperable.

36. It is my further understanding that given the presumption that an allowed patent claim is valid, in order to assert that an allowed patent claim is invalidated by one or more prior art references, it is necessary to show that a

reference (or an appropriate combination of references):

(a) Is (are) properly considered as being prior art to the patent containing the claim, and

(b) Discloses (disclose) each and every limitation of that claim either expressly or inherently.

D. Claim Construction

37. To determine whether the Challenged Claims are valid in light of the prior art, it is necessary to understand the meaning of the various claim terms. I understand that claims of a patent are to be construed based on their claim language, the patent's specification, and the patent's file history. I understand that one also may look at extrinsic evidence to help decipher the meaning and construction of the claims, including, but not limited to, sources such as appropriate dictionaries, the general knowledge of one skilled in the art, treatises, white papers, relevant journals, etc., as long as that extrinsic evidence does not contradict the patent's claims, file history, or specification.

38. I further understand that a patentee may choose to be his/her own lexicographer and define a term differently than the term's plain and ordinary meaning in the art and that, under such circumstances, the patentee's own definition should control. Additionally, a claim term may not be entitled to its plain and ordinary meaning in the art, when the patentee has expressly disclaimed the scope under such plain and ordinary meaning through descriptions in the specifications or

statements made during prosecution of the patent applications.

39. I have been informed that a person of ordinary skill in the art is deemed to read a claim term not only in the context of the particular claim in which the term appears, but also in the context of the entire patent, including the specification, other claims, and prosecution history.

40. I understand that a dependent claim is a claim that incorporates by reference all limitations of its independent claim and of any intervening claims. As a general guideline, the scope of a dependent claim is narrower than that of its independent claim.

41. For purposes of my opinions in this report, I have been asked to assume a priority date of at least May 21, 1998, based on the Israeli Patent Application No. 124571. In other work related to the ’941 patent, I have reviewed Ancora source code productions that, in my professional opinion, establish a priority date of at least March 31, 1997. My analysis of the issues in this proceeding would be the same under either priority date.

42. When I refer to one of ordinary skill in the art in my opinion below, I am referring to one of ordinary skill in the art as of the relevant date set forth above.

43. For ease of reference, below is a reproduction of each of the Challenged Claims of the ’941 patent.

Claim	Language
1[a][1]	A method of restricting software operation within a license for use with a computer including an erasable, non-volatile memory area of a BIOS of the computer, and a volatile memory area; the method comprising the steps of:
1[b]	selecting a program residing in the volatile memory,
1[c]	using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS, the verification structure accommodating data that includes at least one license record,
1[d]	verifying the program using at least the verification structure from the erasable nonvolatile memory of the BIOS, and
1[e]	acting on the program according to the verification.
2	A method according to claim 1, further comprising the steps of: establishing a license authentication bureau.
3	A method according to claim 2, wherein setting up a verification structure further comprising the steps of: establishing, between the computer and the bureau, a two-way data-communications linkage; transferring, from the computer to the bureau, a request-for-license including an identification of the computer and the license-record's contents from the selected program; forming an encrypted license-record at the bureau by

[1] I note that Dr. Wolfe uses a slightly different numbering scheme for multi-part claims. For example, his numbering for Claim 1 includes the parts of Preamble, 1.a, 1.b, 1.c, and 1.d.

Claim	Language
	encrypting parts of the request-for-license using part of the identification as an encryption key; transferring, from the bureau to the computer, the encrypted license-record; and storing the encrypted license record in the erasable non-volatile memory area of the BIOS.
6	A method according to claim 1 wherein selecting a program includes the steps of: establishing a licensed-software-program in the volatile memory of the computer wherein said licensed-software-program includes contents used to form the license-record.
7[a]	A method according to claim 6 wherein using an agent to set up the verification structure includes the steps of:
7[b]	establishing or certifying the existence of a pseudo-unique key in a first non-volatile memory area of the computer; and
7[c]	establishing at least one license-record location in the first nonvolatile memory area or in the erasable, non-volatile memory area of the BIOS.
8	A method according to claim 6 wherein establishing a license-record includes the steps of: forming a license-record by encrypting of the contents used to form a license-record with other predetermined data contents, using the key; and establishing the encrypted license-record in one of the at least one established license-record locations.
9[a]	A method according to claim 7 wherein verifying the program includes the steps of:
9[b]	encrypting the licensed-software-program's license-record contents from the volatile memory area or decrypting the license-record in the erasable, non-volatile memory area of the BIOS, using the pseudo-unique key; and

Claim	Language
9[c]	comparing the encrypted licenses-software-program's license-record contents with the encrypted license-record in the erasable, non-volatile memory area of the BIOS, or comparing the license-software-program's license-record contents with the decrypted license-record in erasable non-volatile memory area of the BIOS.
10	A method according to claim 9 wherein acting on the program includes the step: restricting the program's operation with predetermined limitations if the comparing yields non-unity or insufficiency.
11	A method according to claim 1 wherein the volatile memory is a RAM.
12	The method of claim 1, wherein a pseudo-unique key is stored in the non-volatile memory of the BIOS.
13	The method of claim 1, wherein a unique key is stored in a first non-volatile memory area of the computer.
14	The method according claim 13, wherein the step of using the agent to set up the verification record, including the license record, includes encrypting a license record data in the program using at least the unique key.
16	The method according to claim 13, wherein the step of verifying the program includes a decrypting the license record data accommodated in the erasable second non-volatile memory area of the BIOS using at least the unique key.

44. As the above chart makes clear, each of the Challenged Claims is dependent on claim 1 of the ’941 patent.

45. Below is a table that I understand summarizes the claim construction orders from various district courts.

Term	**Definition**	**Case**
a computer including an erasable, non-volatile memory area of the BIOS of the computer, and a volatile memory area (preamble of claim 1)	this portion of the preamble is limiting	Ex. 1012 at 5 [LG CASE]
a method of restricting software operation within a license ... (preamble of claim 1)	not limiting; the term "license" does not need to be construed	Ex. 1012 at 2 [LG CASE]
acting on the program according to the verification	Plain and ordinary meaning, wherein the step of "acting on the program" may include, but is not limited to, "restricting the program's operation with predetermined limitations, informing the user on the unlicensed status, halting the operation of the program under question, and asking for additional user interactions."	Ex. 1012 at 4 [LG CASE]
All Asserted Claims	The steps of the Claim do not need to be performed in the order recited.	Ex. 1011 at 19-20, 21 [APPLE CASE]
BIOS	An acronym for Basic Input/Output System. It is the set of essential startup operations that run when a computer is turned on, which tests hardware, starts the operating system, and supports the transfer of data among hardware devices.	Ex. 1011 at 8-11, 20 [APPLE CASE]

Term	Definition	Case
BIOS	Plain and ordinary meaning wherein the plain and ordinary meaning is "An acronym for Basic Input / Output System. It is the set of essential startup operations that begin to run automatically when a computer is turned on, which test hardware, starts the operating system, and support the transfer of data among hardware devices"	Ex. 1012 at 2 [LG CASE]
BIOS (Basic Input/Output System)	An acronym for Basic Input/Output System. It is the set of essential startup operations that run when a computer is turned on, which test hardware, starts the operating system, and support the transfer of data among hardware devices	Ex. 1014 at 4 [TCL CASE]
first non-volatile memory area of the computer	plain and ordinary meaning	Ex. 1012 at 5 [LG CASE]
first non-volatile memory area of the computer	a non-volatile memory that is different from the erasable, non-volatile memory of the BIOS	Ex. 1014 at 14-18, 20 [TCL CASE]
license	does not need to be construed	Ex. 1012 at 2 [LG CASE]
license record	A record from a licensed program with information for verifying that licensed program.	Ex. 1011 at 16-17, 20 [APPLE CASE]
license record	data associated with a licensed program with information for verifying that licensed program	Ex. 1012 at 2 [LG CASE]
license record	a record from a licensed program with information for verifying that licensed program	Ex. 1014 at 9-12, 19–20 [TCL CASE]

Term	Definition	Case
license record	a record having information for verifying that licensed program	Ex. 2043 at 9 [TCT Institution Decision]
memory of the BIOS	does not require construction	Ex. 1012 at 3 [LG CASE]
memory of the BIOS	plain and ordinary meaning, i.e., a memory that stores the BIOS	Ex. 1014 at 11-14, 20 [TCL CASE]
non-volatile memory	memory whose data is maintained when the power is removed or voltage is too low	Ex. 1012 at 1 [LG CASE]
non-volatile memory area of the BIOS	memory area of BIOS whose data is maintained when the power is removed	Ex. 1014 at 4 [TCL CASE]
non-volatile memory of the BIOS	does not require construction	Ex. 1012 at 3 [LG CASE]
non-volatile memory	Memory whose data is maintained when the power is removed.	Ex. 1011 at 5-7, 20 [APPLE CASE]
order of steps (claim 1)	Use of the verification structure, as described in Limitation 3, cannot complete until the "set up a verification structure" step has completed, as described in Limitation 2. “Acting on the program according to the verification,” as described in Limitation 4, cannot complete until the “verifying the program” is completed as described in Limitation 3. The “selecting a program residing in the volatile memory” as described in Limitation 1 can occur at any time. (*See also* footnotes on Ex. 1012 at 5)	Ex. 1012 at 5 [LG CASE]

Term	Definition	Case
order of steps (claim 1)	Claim 1 presents restrictions only on when certain actions must complete before another is also completed; therefore the order of claim 1 steps are as follows: Use of the verification structure, as described in Limitation (c), cannot complete until the "set up a verification structure" step has completed, as described in Limitation (b); "acting on the program according to the verification," as described in Limitation (d), cannot complete until the "verifying the program" is completed as described in Limitation (c); the "selecting a program residing in the volatile memory," as described in Limitation (a), can occur at any time[1] [1] Limitation [a] = "selecting a program residing in the volatile memory; Limitation [b] = "using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS, the verification structure accommodating data that includes at least one license record; Limitation [c] = "verifying the program using at least the verification structure from the erasable non-volatile memory of the BIOS," and	Ex. 1013 at 7-10, 37 [LG CASE]

Term	Definition	Case
	Limitation [d] = "acting on the program according to the verification"	
program	a set of instructions that can be executed by a computer	Ex. 1012 at 3 [LG CASE]
selecting a program residing in the volatile memory	plain and ordinary meaning	Ex. 1012 at 3 [LG CASE]
selecting a program residing in the volatile memory	no construction	Ex. 1014 at 20 [TCL CASE]
set up a verification structure	"establishing or certifying the existence of a pseudo-unique key and establishing at least one license-record location"[1] [1] Footnote not for jury. "Establishing at least one license-record location" may include the steps of "forming a license-record by at least partially encrypting the contents used to form a license-record with other predetermined data contents, using at least part of the pseudo-unique key; and storing the encrypted license-record"	Ex. 1012 at 4 [LG CASE]
set up a verification structure	no construction	Ex. 1014 at 20 [TCL CASE]
using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS	Plain and ordinary meaning, wherein the plain and ordinary meaning of "agent" is "a software program or routine"	Ex. 1012 at 3 [LG CASE]

Term	Definition	Case
using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS	plain and ordinary meaning, wherein the plain and ordinary meaning of "agent" is a software program or routine (§112 ¶ 6 does not apply)	Ex. 1013 at 28–36, 37 [LG CASE]
using the key	using a pseudo-unique key	Ex. 1012 at 5 [LG CASE]
verifying the program using at least the verification structure	Confirming whether a program is licensed using at least the verification structure.	Ex. 1011 at 18-19, 21 [APPLE CASE]
verifying the program using at least the verification structure	confirming whether a program is licensed using at least the verification structure	Ex. 1012 at 4 [LG CASE]
volatile memory	Memory whose data is not maintained when the power is removed.	Ex. 1011 at 5–7, 20 [APPLE CASE]
volatile memory	Plain and ordinary meaning wherein the plain and ordinary meaning is "memory whose data is not maintained when the power is removed"[1] [1] Footnote not for the jury. "For the corner case where the hard disk drive is used as virtual RAM, the data is not accessible by normal means after the power is removed."	Ex. 1012 at 2 [LG CASE]
volatile memory	"memory whose data is not maintained when the power is removed," with the exception that "where the hard disk drive is used as virtual RAM, the data is not accessible by normal means after the power is removed"	Ex. 1014 at 4–5, 20 [TCL CASE]

Term	Definition	Case
wherein establishing a license - record includes the steps of: forming a license - record by encrypting of the contents used to form a license - record with other predetermined data contents, using the key; and establishing the encrypted license - record in one of the at least one established license - record locations	no construction	Ex. 1014 at 18–20 [TCL CASE]

E. Abstract Ideas

46. I have been informed and understand that abstract ideas are not patentable. However, I also understand that, at some level, all inventions embody, use, reflect, rest upon, or apply abstract ideas, and that an invention is not removed from patentability simply because an abstract idea is in some way involved.

47. I am aware that the Federal Circuit held in *Ancora Technologies, Inc. v. HTC America, Inc.*, 908 F.3d 1343 (Fed. Cir. 2018) that claim 1 of the ’941 patent—the claim on which the other Challenged Claims depend—satisfies 35 U.S.C. § 101 as a matter of law.

F. The Manner of Rebuttal

48. Unlike the standard for proving infringement, to rebut an assertion of

invalidity, I understand that it is sufficient to show that at least one of the limitations of an allowed claim has not been shown to be found in the prior art. That is what I have done for each Challenged Claim and for each item of allegedly invalidating prior art, and for each combination of items of allegedly invaliding prior art addressed in the Wolfe Declaration. In other words, in rebutting allegations of invalidity, it is not necessary to demonstrate that each and every limitation of a Challenged Claim is missing from the art and/or combinations of art cited in the Wolfe Declaration.

G. The Level of Ordinary Skill in the Art and Person of Ordinary Skill in the Art

49. When interpreting a patent, I understand that it is important to view the disclosure and claims of that patent from the level of a person of ordinary skill in the relevant art (a "POSITA" hereafter) at the time of the invention.

50. My opinion of the level of ordinary skill in the art of the '941 patent is informed by my personal experience working in the fields of computer software, the Internet, and associated technologies, my knowledge of colleagues and others working in those fields as of and for several years before the time frame applicable to the '941 patent, and my study of the patent and its file history.

51. Taking the above factors into account and based on my review of the specification of the '941 patent and my knowledge and experience of the field, I

believe that a POSITA of the ’941 patent would be an individual who, as of the 1997/1998 time frame, had earned an accredited Bachelor’s degree in electrical engineering, computer engineering, or computer science, or an equivalent education or level of knowledge, and two years of industrial experience with computer hardware, software, and operating systems.

52. I understand that Dr. Wolfe stated in his report that “a person of ordinary skill with respect to the subject matter of the ’941 Patent at the time of the alleged invention would have had at least a B.S. degree in computer science, computer engineering, or electrical engineering (or equivalent experience) and would have had at least two years of experience with computer science and computer engineering, including information encryption, computer architecture, and firmware programming. This definition is approximate, and additional educational experience in computer science and computer engineering could make up for less work experience and vice versa.” (Ex. 1003 [Wolfe Declaration] at ¶ 24.)

53. While Dr. Wolfe’s standard is not identical to mine, none of my opinions set forth in this report would be different under his level of ordinary skill in the art.

H. Avoidance of Impermissible Hindsight

54. It is my further understanding that although the *KSR* decision discussed above has led to the elimination of the “teaching, suggestion or motivation” test as

the sole test for judgment whether art can be combined for the purposes of an obviousness assertion, the use of "impermissible hindsight" is still inappropriate when making such an assertion. I note that § 2142 of the Manual of Patent Examining Procedure includes this direction to patent examiners:

> The tendency to resort to "hindsight" based upon applicant's disclosure is often difficult to avoid due to the very nature of the examination process. However, impermissible hindsight must be avoided and the legal conclusion must be reached on the basis of the facts gleaned from the prior art.

I. Requirements for Asserting Obviousness

55. It is my further understanding that when assertions of invalidity are based on a combination of references that allegedly disclose a given patent claim, such assertions must include a discussion of the likelihood that the proposed combinations can be made by those of ordinary skill and, if made, will have a likelihood of success.

56. Section 2143 of the Manual of Patent Examination Procedure states:

> E. "Obvious To Try" – Choosing From a Finite Number of Identified, Predictable Solutions, With a Reasonable Expectation of Success
>
> […]
>
> The rationale to support a conclusion that the claim would have been obvious is that "a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the

> anticipated success, it is likely that product [was] not of innovation but of ordinary skill and common sense. In that instance the fact that a combination was obvious to try might show that it was obvious under § 103." *KSR*, 550 U.S. at 421, 82 USPQ2d at 1397.

57. Section 2143.02 of the Manual of Patent Examination Procedure states:

> I. OBVIOUSNESS REQUIRES A REASONABLE EXPECTATION OF SUCCESS
>
> Where there is a reason to modify or combine the prior art to achieve the claimed invention, the claims may be rejected as prima facie obvious provided there is also a reasonable expectation of success.
>
> […]

58. Obviousness does not require absolute predictability, however, at least some degree of predictability is required.

VI. Summary of the '941 Patent

59. Entitled "Method of Restricting Software Operation within a License Limitation," the abstract of the '941 patent explains that the invention concerns:

> A method of restricting software operation within a license limitation that is applicable for a computer having a first non-volatile memory area, a second non-volatile memory area, and a volatile memory area. The method includes the steps of selecting a program residing in the volatile memory, setting up a verification structure in the non-volatile

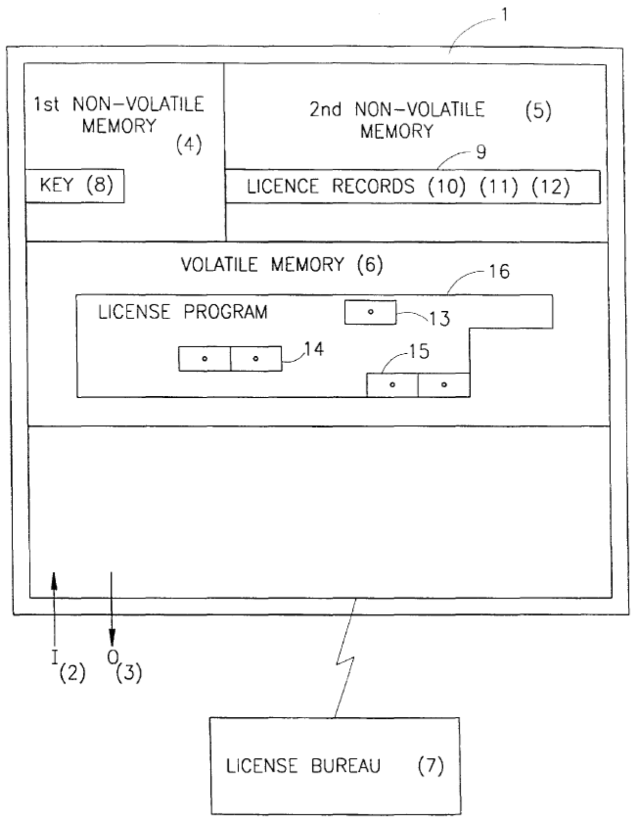
> memories, verifying the program using the structure, and acting on the program according to the verification.

(Ex. 1001 ['941 patent] at Abstract.)

60. Briefly stated another way, the '941 patent teaches a new method for restricting software operation within a license for use and restricting an unauthorized software program's operation. Prior art approaches to the problem required using a hardware attachment (such as a "dongle") or recording license information in an area of a device such as a hard disk drive that is easily accessible to hackers. (*Id.* at 1:19–32.) The '941 patent identifies problems with both of these known methods. First, the software-based products "validate authorized software usage by writing a license signature onto the computer's volatile memory (e.g., hard disk)." (*Id.* at 1:19–21.) But the '941 patent explains that the license signature is vulnerable to "hackers" and to "physical instabilities" of the hardware. (*Id.* at 1:21–26.) Second, the hardware-based products required "a dongle that is coupled e.g. to the parallel port of the P.C." (*Id.* at 1:27–29.) But the '941 patent explains that dongles are "expensive, inconvenient, and not particularly suitable for software that may be sold by downloading (e.g., over the internet)." (*Id.* at 1:29–32.)

61. In the '941 patent, the inventors teach storing license records identifying and enabling the use of software in an area separate and apart from the area normally accessible to programs running on the computer, in BIOS storage.

Specifically, the ’941 Patent discloses using existing computer hardware with “a conventional BIOS module in which a key was embedded at the ROM section thereof.” (*Id.* at 1:45–47.) The BIOS module further includes “another (second) non-volatile section of the BIOS, e.g., E^2PROM (or the ROM)” in which “memory may optionally be erased or modified (using E^2PROM manipulation commands).” (*Id.* at 1:66–2:4.) FIG. 1 of the ’941 patent illustrates a first non-volatile memory 4 (e.g., ROM section of the BIOS) including a key 8, and a second non-volatile memory 5 (e.g., E^2PROM section of the BIOS) with a license-record area 9 that includes license records 10–12. (Ex. 1001 at 5:10–27.)


1
1st NON-VOLATILE MEMORY (4)
KEY (8)
2nd NON-VOLATILE MEMORY (5)
9
LICENCE RECORDS (10) (11) (12)
VOLATILE MEMORY (6)
16
LICENSE PROGRAM
13
14
15
I (2)
O (3)
LICENSE BUREAU (7)

(Ex. 1001 at FIG. 1.)

62. BIOS, at the time of the invention, was understood to be "the set of essential software routines that test hardware at startup, start the operating system, and support the transfer of data among hardware devices." (*See, e.g.*, Ex. 2009 [Microsoft Press Computer Dictionary (3rd Ed.)].) BIOS memory therefore was typically used for storing code and other configuration data that assisted in the start-up and operation of a computer. Although BIOS served these critical functions, it was "usually invisible to computer users." (*Id.*) BIOS determines how an operating

system is launched, and importantly, which operating systems it is willing to launch. (Ex. 2009 [Microsoft Press Computer Dictionary (3rd Ed.)]; Ex. 2044 [Free On-Line Dictionary of Computing (June 6, 1999)].) A BIOS may, for example, be configured to launch only operating systems that it recognizes as ones that will not directly change certain security-sensitive areas of the device's non-volatile memory but instead will consult BIOS in order to do so. In the context of the '941 patent, the courts have construed BIOS to be

> An acronym for Basic Input/Output System. It is the set of essential startup operations that run when a computer is turned on, which test hardware, starts the operating system, and supports the transfer of data among hardware devices.

(*See*, *e.g.*, Ex. 1011 at 8–11, 20; Ex. 1012 at 2; Ex. 1014 at 4.)

63. The '941 patent explains that more than one non-volatile memory may be found in typical computers known in the art. "Today a processor normally includes a first non-volatile memory, a second non-volatile memory, and data linkage access to a volatile memory." (Ex. 1001 at 3:21–24.) As described in the '941 patent, the computer includes a ROM section of a BIOS and an E^2PROM section of the BIOS. Persons skilled in the art would understand ROM to mean a chip with contents that cannot be modified. In contrast, an E^2PROM was a known type of chip capable of being rewritten using electrical signals.

64. The '941 patent explains that a license record is created in the second

non-volatile section of the BIOS during "an initial license establishment procedure" that sets up a "verification structure" using E^2PROM manipulation commands that enable the method to "add, modify or remove licenses" in the BIOS. (*See id.* at 1:33–2:9.) In one example, setting up the verification structure includes "establishing at least one license-record location in the first or second nonvolatile memory area." (*Id.* at 6:20–21.) License records can then be created and stored in the verification structure. With reference to FIG. 1, the '941 patent explains that license record fields (13–15) appended to a licensed program 16 are encrypted to form license records (10–12). (*Id.* at 5:27–43.) The key stored in the ROM portion of the BIOS is used to encrypt the license records, such that the license record is rendered useless if copied to a second computer with a different key. (*See id.* at 2:27–59; 6:22–26.) The encrypted license records are then established in the license record locations (*e.g.*, 10–12 in FIG. 1). (*Id.* at 6:26–28.)

65. By storing license records in the BIOS, outside the normal reach of the operating system, the '941 patent explains that it is much harder for an attacker (described in the patent as encompassing a "hacker," a "skilled system's [*sic*] programmer," an "ordinary software hacker," and a "professional-like hacker") to employ ordinary operating system programs to tamper with a computer's operation and/or install unauthorized software. (Ex. 1001 at 3:4–17.) As the '941 patent explains, "An important advantage in utilizing non-volatile memory such as that

residing in the BIOS is that the required level of system programming expertise that is necessary to intercept or modify commands, interacting with the BIOS, is substantially higher than those needed for tampering with data residing in volatile memory such as hard disk." (*Id.* at 3:4–9.)

66. In some examples, the method may rely on a "license bureau (7)" that is linked to the computer as illustrated in FIG. 1. (*Id.* at 5:17–18.) Establishing the license record is a task that "may be shared between the bureau and the computer, done exclusively at the computer, or done exclusively at the bureau." (*Id.* at 4:33–35.)

67. The '941 patent describes an example of verifying the license that employs a license verifier application running in the computer, which "accesses the program under question, retrieves therefrom the license record, encrypts the record utilizing the specified unique key (as retrieved from the ROM section of the BIOS) and compares the so encrypted record to the encrypted records that reside in the E^2PROM." (*Id.* at 2:10–19; 6:29–39.) The program is allowed to run if the encrypted records match, or is halted or otherwise interrupted if they do not. (*Id.* at 2:19–26; 6:40–52.)

68. The Federal Circuit has also twice identified at least the following novel aspects of the '941 patent, including by describing "[t]he inventive method" of the '941 patent as: "us[ing] a modifiable part of the BIOS memory—not other computer

memory—to store the information that can be used, when a program is introduced into the computer, to determine whether the program is licensed to run on that computer. BIOS memory is typically used for storing programs that assist in the start-up of a computer, not verification structures comparable to the software-licensing structure embodied by the claimed invention. Using BIOS memory, rather than other memory in the computer, improves computer security, the patent indicates, because successfully hacking BIOS memory (i.e., altering it without rendering the computer inoperable) is much harder than hacking the memory used by the prior art to store license-verification information." *Ancora Techs., Inc. v. HTC Am., Inc.*, 908 F.3d 1343, 1345 (Fed. Cir. 2018).

69. As the Federal Circuit has also observed, "[t]he prosecution history reinforces what the patent itself indicates about the change in previous verification techniques for computer security"; indeed, the USPTO Examiner's Reasons for Allowance of the '941 patent emphasized the patent's solution of "using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS." *Id.* at 1349 (quotation marks omitted).

70. As the applicants for the '941 patent explained, "BIOS is a configuration utility. Software license management applications, such as the one of the present invention, are operating system (OS) level programs. Therefore, BIOS programs and software licensing management applications do not ordinarily interact

or communicate because when BIOS is running, the computer is in a configuration mode, hence OS is not running. Thus, BIOS and OS level programs are normally mutually exclusive." (Ex. 2011 ['941 patent file history] at ANCC000153.) The person skilled in the art would have recognized BIOS code as separate from the operating system on a given computer. As described in the Barron's definition of BIOS, for example, using BIOS to handle input-output functions was "important because if the hardware is changed (for example by installing a newer kind of video adapter) the BIOS can be changed to match it, and there is no need to change the application programs." (*See* Ex. 2042 [Barron's Dictionary of Computer and Internet Terms (5th Ed. 1996)].) One benefit of keeping BIOS code separate from the operating system was therefore to allow the operating system (and associated application software) to operate on a variety of different hardware configurations. (*Id*.)

71. Further, the applicants distinguished prior art (including prior art BIOS) by explaining, for example, that

> [T]he present invention proceeds against conventional wisdom in the art. Using BIOS to store application data such as that stored in [a] local cache for licenses is not obvious. The BIOS area is not considered a storage area for computer applications. An ordinary skilled artisan would not consider the BIOS as a storage medium to preserve application data for at least two reasons.

> First, OS does not support this functionality and is not recognized as a hardware device like other peripherals. Every OS provides a set of application program interfaces (APIs) for applications to access storage devices such as hard drives, removable devices, etc. An ordinary person skilled in the art makes use of OS features to write dat[a] to storage mediums. There is no OS support whatsoever to write data to the system BIOS. Therefore, an ordinary person skilled in the art would not consider the BIOS as a possible storage medium. Furthermore, it is common that all peripheral devices in the PC are listed and recognized by the OS except for the BIOS. This supports the fact that the BIOS is not considered a peripheral device. Accordingly, an ordinary person skilled in the art would not consider the BIOS for any operation, including writing to the BIOS.
>
> Second, no file system is associated with the BIOS. Every writable device connected to the PC is associated with an OS file system to arrange and manage data structures. An example for such a file system would be FAT, FAT32, NTFS, HPFS, etc. that suggests writing data to the writable device. No such file system is associated with the BIOS. This is further evidence that OS level application programmers would not consider the BIOS as a storage medium for license data.

(Ex. 2011 ['941 patent file history] at ANCC000154.) In this excerpt, the applicants described some aspects of BIOS that were familiar to practitioners and that would have contributed to practitioners' approaches to system design even as BIOS continued to evolve.

72. Persons skilled in the art were familiar with the ways in which operating systems differed from BIOS. For example, the Ginter reference cited by the Examiner explains the operating system as follows:

> To organize the CPU's execution capabilities with available RAM, ROM and secondary storage devices, and to provide commonly used functions for use by programmers, a piece of software called an "operating system" is usually included with the other components. Typically, this piece of software is designed to begin executing after power is applied to the computer system and hardware diagnostics are completed. Thereafter, all use of the CPU, main memory and secondary memory devices is normally managed by this "operating system" software. Most computer operating systems also typically include a mechanism for extending their management functions to I/O and other peripheral devices, including commonly used functions associated with these devices.
>
> By managing the CPU, memory and peripheral devices through the operating system, a coherent set of basic functions and abstraction layers for hiding hardware details allows programmers to more easily create sophisticated applications. In addition, managing the computer's hardware resources with an operating system allows many differences in design and equipment requirements between different manufacturers to be hidden. Furthermore, applications can be more easily shared with other computer users who have the same operating system, with significantly less work to support different manufacturers' base hardware and peripheral devices.

(Ex. 2019 [Ginter] at 82:27–52.) The person having skill in the art would recognize the "hardware diagnostics" described above as being among the tasks performed by BIOS software upon startup of the computer.

73. Examples of operating systems that were also known at the time of the invention included Windows NT, Windows 95, DOS, Macintosh OS, and UNIX-based operating systems. (Ex. 2014 [Misra] at 5:67–6:5; Ex. 2019 [Ginter] at 81:35-36.) Each of these commercially available operating systems relies on a component that begins to run automatically when the computer is turned on to start the operating system (often referred to "booting" the computer).

74. The USPTO Examiner agreed with the applicants that the inventors' invention was neither anticipated nor rendered obvious by the prior art. As the Examiner summarized in the Reasons for Allowance of the '941 patent, "the key distinction between the present invention and the closest prior art, is that the [prior art] systems . . . run at the operating system level and BIOS level, respectively. More specifically, the closest prior art systems, singly or collectively, do not teach licensed programs running at the OS level interacting with a program verification structure stored in the BIOS to verify the program using the verification structure and having a user act on the program according to the verification. Further, it is well known to those of ordinary skill of the art that a computer BIOS is not setup to manage a software license verification structure. The present invention overcomes this

difficulty by using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS." (Ex. 2011 ['941 patent file history] at ANCC000162.)

VII. '941 Patent Prosecution History

75. Dr. Wolfe does not consider the file history of the '941 patent or the references cited during prosecution. While examining the application that eventually issued as the '941 patent, the examiner rejected the then-pending claims on grounds that relied on various combinations of four references:

- U.S. Patent No. 5,892,900 by Ginter et al. (hereinafter "Ginter")
- U.S. Patent No. 5,684,951 by Goldman et al. (hereinafter "Goldman")
- U.S. Patent No. 6,189,146 to Misra et al. (hereinafter "Misra")
- U.S. Patent No. 5,479,639 to Ewertz et al. (hereinafter "Ewertz")

I also understand that the applicant submitted a whitepaper describing a proposed commercial embodiment of the invention disclosed in the '941 patent, titled: Beeble Hardware Stamping^{TM} Technology Overview. I discuss each of these references below and summarize relevant portions of the examiner's actions and the responses filed by the applicant.

A. Ginter

76. Ginter discloses "a distributed virtual distribution environment (VDE) that may enforce a secure chain of handling and control, for example, to control

and/or meter or otherwise monitor use of electronically stored or disseminated information." (Ex. 2019 [Ginter] at Abstract; *see also id.* at 2:20–32, 6:29–67.) Software application publishers are among the content providers Ginter discloses as users of the disclosed VDE. (*Id.* at 14:1–6.)

77. Ginter explains that "tamper resistance and concealment of VDE control process execution and related data storage activities" is used to improve the overall security of the VDE system. (*Id.* at 21:22–33.) Ginter discloses an "electronic appliance" with a secure processing unit (SPU) capable of operating within the VDE. (*Id.* at 60:7–15.) Ginter explains that, in its preferred embodiment, the VDE system uses "special purpose tamper resistant Secure Processing Units (SPUs) to help provide a high level of security for VDE processes and information storage and communications." (*Id.* at 3:67–4:4, 20:47–59.)

78. Ginter's preferred SPU was a non-standard piece of hardware providing the capabilities disclosed by Ginter. (*See id.* at FIG. 6.) Ginter repeatedly describes the SPUs as "special purpose" hardware devices. (*Id.* at 3:67–4:4, 20:47–53, 22:14–25, 60:40–44, 67:48–51, 88:37–43.) Further, in preferred embodiments the SPU is enclosed using "special purpose semiconductor packaging techniques." (*Id.* at 21:25–33.) The use of separate special purpose hardware contrasts with examples in which Ginter hypothesizes that VDE capabilities could be integrated into, for example, a "standard microprocessor," or a "general purpose processor." (*Id.* at

21:1–21, 60:49–52.)

79. In an example illustrated in FIG. 8, Ginter describes an electronic appliance 600 such as a computer, comprising one or more SPU 500. (*Id.* at 62:31–63, 63:67–65:15.) Alternatively, "SPU 500 may be integrated together with one or more other CPU(s) (e.g., a CPU 654 of an electronic appliance) in a single component or package." (*Id.* at 65:21–24.) The SPU, as illustrated in Ginter's FIG. 9, "may comprise a combination of a masked ***ROM 532a*** and an ***EEPROM and/or equivalent "flash" memory 532b***." (*Id.* at 70:39–71:15 (emphasis added), FIG. 9.)

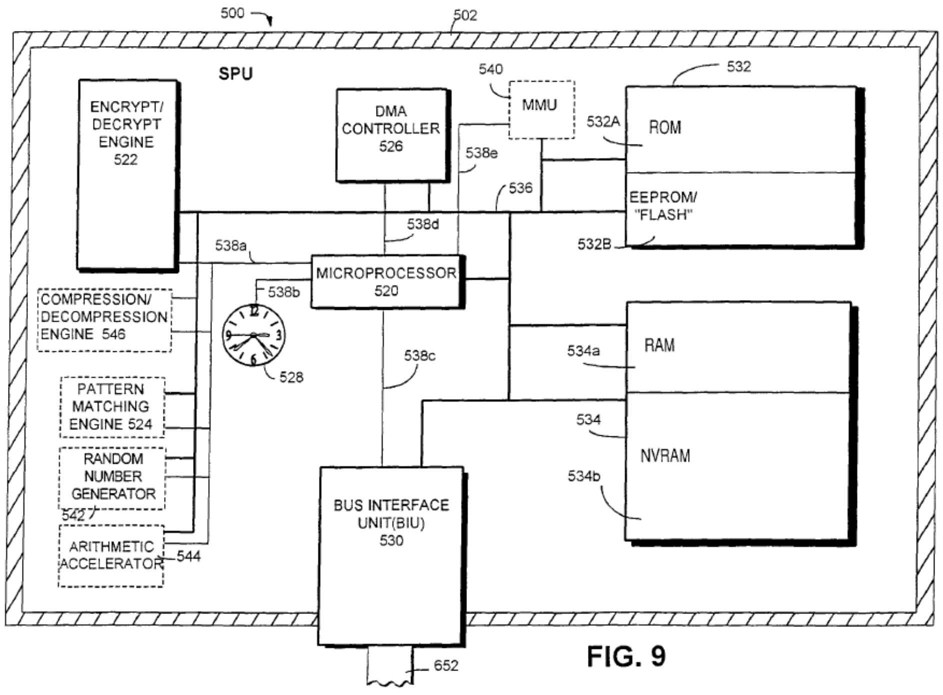


FIG. 9

(*Id.* at FIG. 9.)

80. Ginter discloses determining a "signature" comprising unique

information about the electronic appliance 600. (*Id.* at 239:15–25.) In one example, Ginter explains that "an otherwise unused section of the non-volatile CMOS RAM 656a may be used to store a signature 3497d." (*Id.* at FIG. 69G, 239:50–53.)

81. The examiner cited Ginter in a rejection in the December 20, 2000 office action. (Ex. 2011 at ANCC000077–83.) The applicant argued that Ginter did not disclose, among other things, setting up a verification structure and verifying the program using the verification structure. (*Id.* at ANCC000093–97.) Applicant argued that Ginter required "add-on hardware" that was "not part of the PC." (Ex. 2011 at ANCC000095.) As the applicant explained in detail:

> [I]n col. 70, line 23 – col. 71, line 25 Ginter et al. describe the architecture as add-on hardware which is named "SPU". . . . Col. 64, lines 16-21 explicitly detail[s] the fact that the SPU is a hardware, add-on not part of the PC.

(*Id.*) The applicant emphasized this point again when explaining:

> There is no mention whatsoever in Ginter et al. . . . referred to by the Examiner of a process where a software program verifies its authenticity using a license (verification structure) stored in the second volatile non-volatile memory. The functionality described in these portions of Ginter et al. is the different functionality that add-on hardware, referred to as SPU, can perform.

(*Id.* at ANCC000096.)

82. The examiner maintained the same rejections, again citing Ginter, in

the June 22, 2001 office action. (*Id.* at ANCC000106–08, ANCC000110–16.) In his discussion of Ginter, the examiner cited portions of Ginter's disclosure that describe "hidden secret storage." (Ex. 2011 at ANCC000111 (citing Ex. 2019 [Ginter] at 245:55–246:24).) For example, Ginter discloses that sensitive information may be protected by "cryptography employing keys and/or authentication values hidden in normally inaccessible locations in the appliance 600," including: "[n]on-volatile, writable, storage in the appliance or its components, such as that used for . . . standard BIOS software, etc." (Ex. 2019 [Ginter] at 245:55–246:24.)

83. The applicant responded on November 16, 2001 by amending the claims, specifically amending claim 1 to recite "<u>using an agent to</u> set up verification structure in the <u>erasable</u>, non-volatile memory <u>of the BIOS</u>, the <u>verification</u> structure accommodat<u>ing</u> data that includes at least one license record." (*Id.* at ANCC000130 (additions emphasized).)

84. As discussed further below, the examiner ultimately allowed the amended claims, distinguishing Ginter because Ginter does not teach utilizing BIOS as the non-volatile means for storing a licensed software verification structure, and does not teach licensed programs running at the OS level interacting with a program verification structure stored in the BIOS. (*Id.* at ANCC000161–62.)

B. Goldman

85. Goldman discloses a system for "user validation with respect to a multi-

user computer system." (Ex. 2045 [Goldman] at 1:8–10.) In this system, a computer system 112 is connected via an internet interface 110 to an application system 310. (*Id.* at FIG. 1, 4:59–5:4.) As shown in Figure 3A, the system provides a user validation system 310a communicating with the computer system 112 via the user's email account 220 and elements of the internet interface 110. (*Id.* at 5:34–40.)

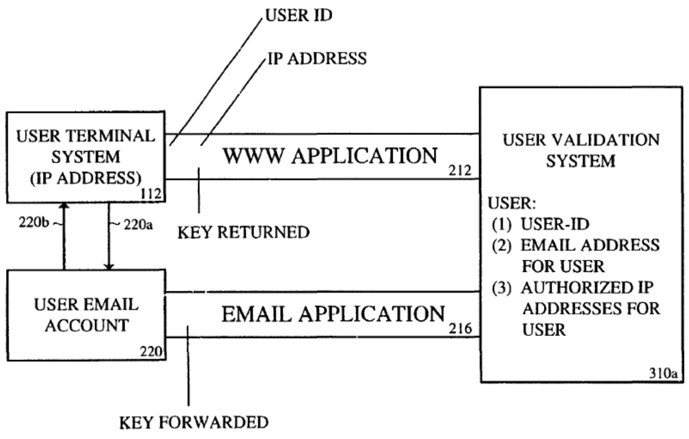


FIG. 3A

(*Id.* at FIG. 3A.)

86. The remote user validation system 310a maintains a database having an entry for each authorized user, including "the user's identification (userID), the user's email address, and each IP address for which the user is authorized." (*Id.* at 6:12–16.) In certain situations, the validation system generates a "pseudo unique" validation key from the userID, the current IP address, and a secret code, which can be used to verify the user when the user is logged in from a new IP address. (*Id.* at

Abstract, 8:42–54, 9:26–33.)

87. During prosecution of the ’941 patent the Examiner rejected certain claims, citing Goldman in combinations that incorporated Goldman’s pseudo unique key. (*E.g.*, Ex. 2011 at ANCC000080–83, ANCC000113–16, ANCC000143–45.) I understand that no issues related to Goldman have been raised in the pending IPR.

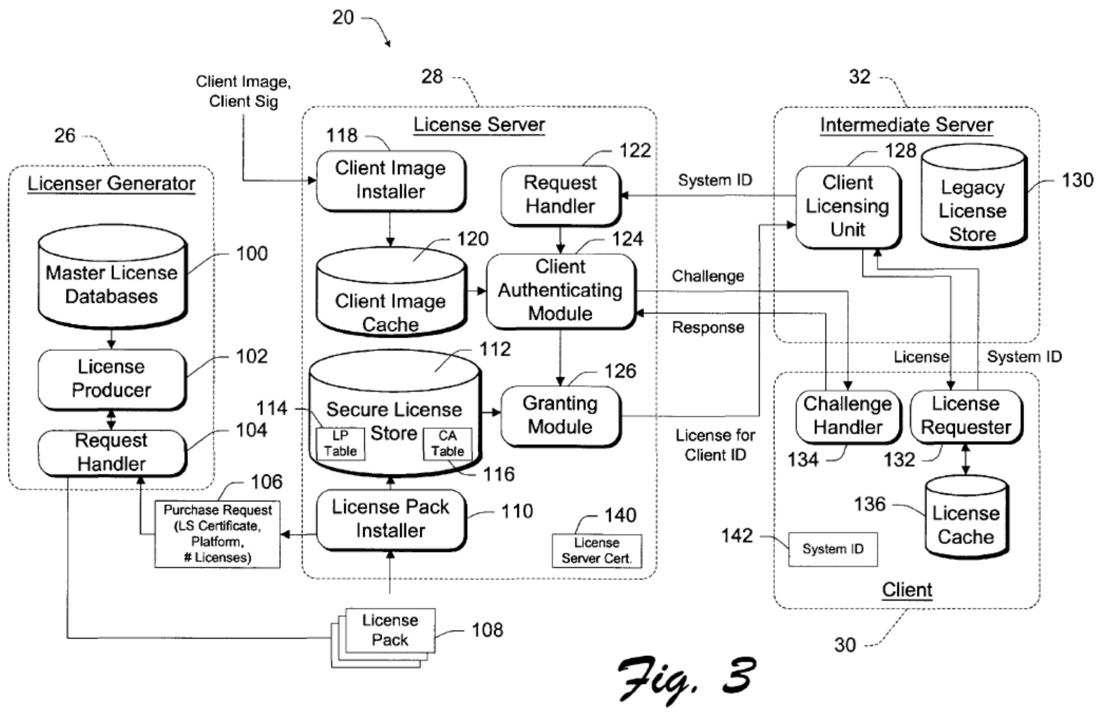
C. Beeble Hardware Stamping Technology™ Overview

88. The applicant’s November 16, 2001 response notes that “a description of a specific embodiment of the invention is attached hereto.” (*Id.* at ANC000129.) The file history includes a white paper authored by Miki Mullor, one of the inventors of the ’941 patent, titled “Beeble Hardware Stamping™ Technology Overview.” (*Id.* at ANC000178–85.) The white paper explains that “[i]n order for a user to access the BIOS EEPROM[,] ***proprietary software***” needed to be developed. (*Id.* at ANC000180 (emphasis added).) The document explains that the “Beeble License Manager is a windows control panel applet that allows users to self manage Beeble licenses installed on their machine.” (*Id.* at ANC000182.) The control panel applet was responsible for “installation and removal of Beeble licenses.” (*Id.*) A person skilled in the art would have recognized the described control panel applet as an example of an OS-level software program or routine (“agent”; *see* Ex. 1012 at 3).

D. Misra

89. Misra discloses an invention that “relates to a system and method for

enforcing software licenses." (Ex. 2014 [Misra] at 1:7–8.) Misra's system includes an architecture with four components: license generator 26, license server 28, intermediate server 32, and client 30, detailed below in FIG. 3. (*Id.* at 6:21–24.) Misra explains that "[b]ecause the clients might not have network connectivity to the license server 28, one or more intermediate servers 32(1) and 32 (2) [shown in Figure 1], can act as an intermediary for the clients." (*Id.* at 4:31–34.)



(*Id.* at FIG. 3.) Misra explains that the clients 30 can be "many different kinds of computers, including a desktop personal computer, a workstation, a laptop, a notebook computer, a handheld PC, and so forth;" or a "terminal device, which is a low cost machine." (*Id.* at 5:13–22.) Misra illustrates an example implementation of a computer 40, which can be used to implement the clients. (*Id.* at FIG. 2, 5:26–28.) As shown in Figure 2, the computer 40 includes a read only memory (ROM) 48, and a "basic input/output system 52 (BIOS) is stored in ROM 48. (*Id.* at 5:37–39.)

90. Misra describes how client software licenses are verified each time the software is run. "Prior to working with the client and providing access to files, the intermediate server 32 wants to verify first that the client has a valid software license issued by a recognized license server. The client 30 may or may not have a valid license, So the intermediate server makes an initial evaluation when the client attempts to connect. Generally, if the client 30 has a valid license, the client is permitted to connect and use the server's resources. If the client 30 offers an invalid license, the client is disconnected." (*Id.* at 14:2-11.)

91. As shown in FIG. 3, the client 30 includes a client system ID 142 that "is a unique identifier of the client computer. (*Id.* at 12:50–51.) Misra discloses that

> the system IDs can be based on information collected form a computer's hardware and installed software. For example, hard disk volume numbers, network cards, registered software, video cards, and some microprocessors contain unique identifiers. On PCs, this information can be combined to uniquely identify a particular PC. Other information that might be used includes total RAM and floppy disk drive configuration.

(*Id.* at 12:56–63.) The client ID is used to verify a client's license at the intermediate server 32, which checks the client ID for a given client against the client's license "by extracting the client ID from the license . . . and comparing it to the client ID received from the client. If the two match, the client ID passes." (*Id.* at 14:34–38.) In short, the client ID is included in the license to "prevent the software license from being copied from one client machine to another." (*Id.* at 15:29–32.)

92. Misra also describes a license ID, which is a unique identifier assigned to a software license when the software license is issued to a client device. (*Id.* at 11:9–12.) The license ID may be stored within an X.509 certificate indicating the right to use the particular software at issue. (*Id.* at 10:60–67.)

93. The license server 28 in Misra's system incorporates both the client system ID 142 and the license ID in a software license with contents shown in Misra's Table 5. (*Id.* at 10:60–11:24.) Misra further explains that the client 30 (using its license requestor 132) stores the licenses in a license cache 136, which "is kept in persistent (non-volatile) storage." (*Id.* at 12:8–16.) Misra does not, however, teach constructing its license records within the BIOS of the computer, as was noted by the examiner. (Ex. 2011 at ANCC000145.)

94. The examiner cited Misra in a rejection in the January 15, 2002 office action, rejecting then-pending claims 1–23 as being obvious over Misra, Goldman, and Ewertz (discussed below). (Ex. 2011 at ANCC000143–45.) The examiner asserted that Misra discloses many elements recited in the claim. For example, the

examiner equated Misra's programs with the claimed agent, asserting that "Misra et al. also teach . . . programs ('agent') used in the license collation process that belong to various parties". (*Id.* at ANCC000144–45.) The examiner noted, however, that Misra does not teach "constructing license records within a computer BIOS." (Ex. 2011 at ANCC000145.) The examiner relied on Ewertz for the teaching of expanding BIOS memory to store identification or configuration data such as software licenses. (*Id.*)

95. In a January 15, 2002 response to the office action, applicants similarly noted that "Misra fails to teach using the BIOS of a computer to store the license ID. (*Id.* at ANCC000151.) Applicants argued that Misra's system ID "uniquely identifies a computer and simply does not correspond [to] . . . the license information of the present invention as defined by claims 1 and 20." (*Id.* at ANCC000152–53.)

96. In the subsequent notice of allowance, the examiner agreed that neither Ginter nor Misra "teaches utilizing BIOS as the non-volatile means for storing a licensed software verification structure." (*E.g.*, Ex. 2011 at ANCC000161–62.) The examiner further distinguished the art because it failed to teach "licensed programs running at the OS level interacting with a program verification structure stored in the BIOS," as discussed further below. (*E.g.*, *id.* at ANCC000162.)

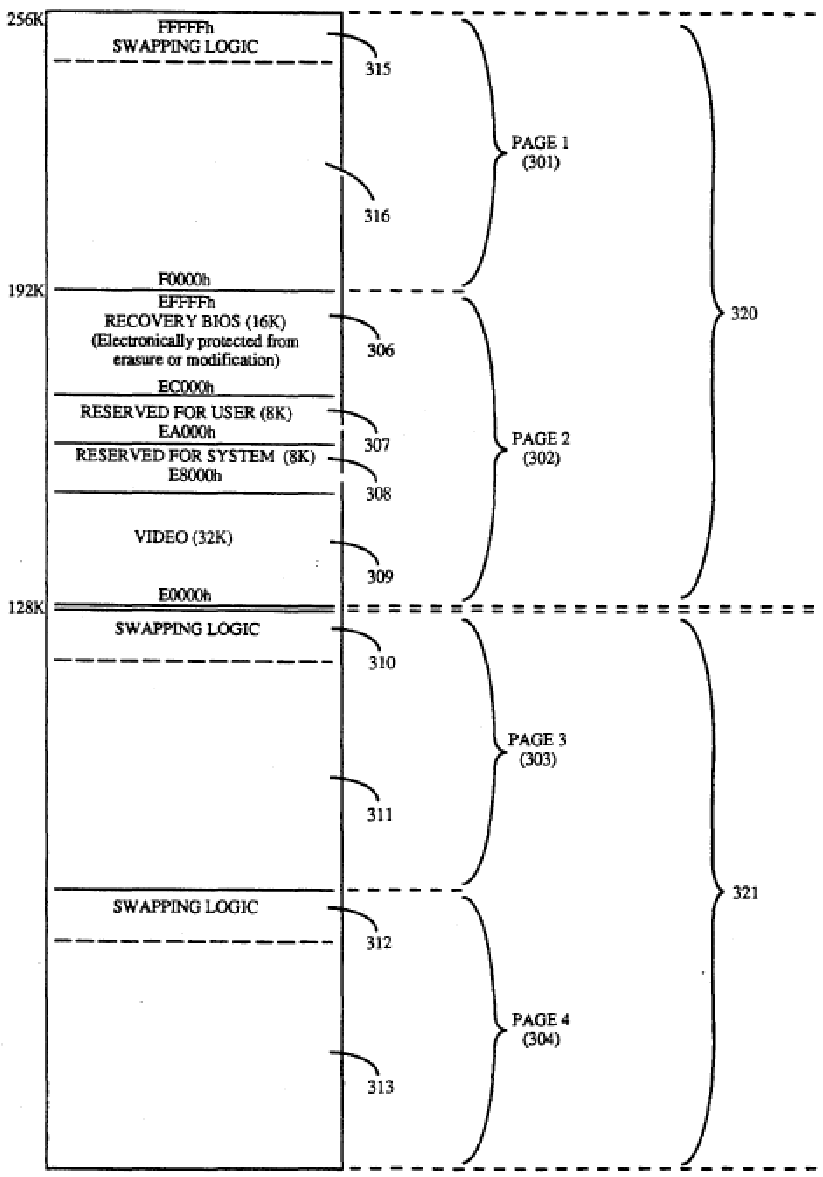
E. Ewertz

97. Ewertz discloses "a paging technique . . . used to expand the usable

non-volatile memory capacity beyond a fixes address space limitation." (Ex. 2015 [Ewertz] at 2:24–26.) Ewertz explains that "Prior [art] versions of the IBM PC use read only memory devices for storage of firmware or a basic input/output system (BIOS) software program." (*Id.* at 1:36–38.) Ewertz explains that "ROM devices with a BIOS contained therein are typically constrained to a specific address range within the address space available." (*Id.* at 1:56–59.) In the IBM PC AT architecture, for example, the ROM BIOS cannot exceed 128K of ROM space. (*Id.* at 1:62–67.) According to Ewertz, developments at the time made it "increasingly unfeasible to fit all desired BIOS features within the 128K boundary of the IBM PC AT architecture." (*Id.* at 2:6–12.)

98. Ewertz illustrates the preferred embodiment of a paged flash memory 103.

FIGURE 2



(*Id.* at FIG. 2.) Ewertz explains that "the BIOS is constrained to the upper 128K of the first Mbyte of the addressable memory space in the computer system." (*Id.* at 5:15–17.) The 128K region 320 is further subdivided, both in prior art and the inventive device.

> The upper region 301 is used for storage of the normal system BIOS while the lower region 302 is used for storage of other logic and data such as overflow BIOS code and/or data, video or other BIOS's, set-up code or data, and other information or logic.

(*Id.* at 5:20–25.) Ewertz further describes a technique for swapping "pages" of the flash memory. Specifically, "Page 3 (303) and Page 4 (304) may be individually swapped into the address space occupied by Page 1 (301). In the preferred embodiment, Page 2 (302) is held static and thus is not used as a swap area." (*Id.* at 3:3–5, 5:30–38.) One of the objects of Ewertz's invention is "to provide a means for maintaining at least one static page." (*Id.* at 3:33–35.)

99. The person skilled in the art would recognize several advantages, identified in Ewertz, for maintaining a static page within the flash memory. First, storing system identification numbers, including "unique serial numbers, printed board assembly (PBA) numbers or operating system license numbers," requires a memory location that cannot be overwritten or otherwise changed. (*See id.* at 12:8–14.) Second, the static page can be used as a backup, in case battery-powered CMOS RAM fails. (*Id.* at 10:14–20.) CMOS memory was often used to store certain configuration information, as discussed in Ewertz. (*Id.* at 9:62–65.)

100. Ewertz discloses that "identification information may be stored in a page of non-volatile memory." (*Id.* at 3:15–17.) This identification information may

include an Ethernet address, system serial numbers, or software license numbers. (*Id.* at 3:18–22.)

101. Ewertz also discloses processing logic for updating the flash memory device, illustrated in Figure 8. (*Id.* at FIG. 8, 10:21–26.) Ewertz explains, however, that this processing logic "resides in the system BIOS." (*Id.* at 10:26–30.) Doing so allows the update routines "to accommodate hardware specific operations with non-volatile memory in particular computer systems." (*Id.* at 10:45–48.) A person having ordinary skill in the art reading Ewertz would understand that writing to the flash memory, as shown in Ewertz's Figure 8, would be done using software running as part of the BIOS code. Ewertz explains that the steps in Figure 8 are started when the BIOS is running. After block 812 at the end of the procedure, "[n]ormal BIOS processing" is resumed. (*Id.* at 11:12–13.)

102. As mentioned above, the examiner cited Ewertz in a rejection in the January 15, 2002 office action, rejecting then-pending claims 1–23 as obvious over the combination of Misra, Goldman, and Ewertz. (Ex. 2011 at ANCC000143–45.) As mentioned above, the examiner relied on Ewertz for its teaching of expanding BIOS memory to store identification or configuration data such as software licenses. (*Id.*) Thus, Ewertz allegedly filled one of the gaps in Misra's disclosure relative to the then-pending claims. (*Id.*) The examiner asserted that, because Ewertz teaches "expanding non-volatile memory (e.g. BIOS) for maintaining data such as software

licenses," it would have been obvious "to use the BIOS to store licenses in the Misra et al. system as they teach of users storing license data in persistent-non-volatile storage." (*Id.* at ANCC000145 (internal citations omitted).)

103. In a January 15, 2002 response to the office action, applicants argued that Ewertz discloses storing identification information such as software licenses numbers "in a non-writeable, non-erasable area of the BIOS during manufacture." (*Id.* at ANCC000152.) As with Misra, applicants also argued that Ewertz's identification information is a static data structure that "uniquely identifies a computer and simply does not correspond [to] . . . the license information of the present invention as defined by claims 1 and 20." (*Id.* at ANCC000152–53.)

104. Further, applicants argued that the person skilled in the art would not have combined Misra and Ewertz. Applicants pointed out differences between the license management application disclosed by Misra and the BIOS functionality disclosed in Ewertz:

> . . . BIOS is a configuration utility. Software license management applications, such as the one of the present invention, are operating system (OS) level programs. Therefore, BIOS programs and software licensing management applications do not ordinarily interact or communicate because when BIOS is running, the computer is in a configuration mode; hence OS is not running. Thus, BIOS and OS level programs are normally mutually exclusive.

> Ewertz teaches that writing to the BIOS area is performed by the BIOS routines:
>
> > "Referring to Fig. 8, processing logic for updating the flash memory device with configuration data, such as EISA information, is illustrated . . .. The processing logic shown in Fig. 8 resides in the system BIOS of the preferred embodiment" Col. 10, lines 20-28
>
> Misra teaches a licensing system that is OS level based:
>
> > "The license generator 26, license server 28 and intermediate server 32 are preferably implemented as computer servers, such as Windows NT servers that run Windows NT server operating systems from Microsoft corporation or UNIX-based servers" Col 5, lines 3-7
>
> Thus, the systems described in Misra and Ewertz are an OS program and a BIOS program, respectively, that cannot run at the same time. Therefore, there is no teaching or suggestion to combine these programs. In fact such a combination would change the operation of the programs, which is an indicia of non-obviousness, see MPEP Sec. 2141.03 and related case law.

(*Id.* at ANCC000153–54.) I agree that persons having ordinary skill in the art would recognize these as critical distinctions.

105. The applicants also argued that the invention "proceeds against conventional wisdom in the art." Specifically, applicants argued that "[t]he BIOS area is not considered a storage area for computer applications." (*Id.* at

ANCC000153–54.) Applicants identified two reasons why the ordinarily skilled artisan would not look to the BIOS memory as a storage medium:

> First, OS does not support this functionality and is not recognized as a hardware device like other peripherals. Every OS provides a set of application program interfaces (APIs) for applications to access storage devices such as hard drives, removable devices, etc. An ordinary person skilled in the art makes use of OS features to write date to storage mediums. There is no OS support whatsoever to write data to the system BIOS. Therefore, an ordinary person skilled in the art would not consider the BIOS as a possible storage medium. Furthermore, it is common that all peripheral devices in the PC are listed and recognized by the OS except for the BIOS. This supports the fact that the BIOS is not considered a peripheral device. Accordingly, an ordinary person skilled in the art would not consider the BIOS for any operation, including writing to the BIOS.
>
> Second, no file system is associated with the BIOS. Every writable device connected to the PC is associated with an OS file system to arrange and manage data structures. An example for such a file system would be FAT, FAT32, NTFS, HPFS, etc. that suggests writing data to the writable device. No such file system is associated with the BIOS. This is further evidence that OS level application programmers would not consider the BIOS as a storage medium for license data.

(*Id*.) I agree that the person having ordinary skill in the art would not have considered the BIOS as an available storage medium for license data from the OS level at the priority date of the ’941 patent.

106. The examiner agreed that the applicants had distinguished the allowed ’941 patent claims from the combination of Misra, Goldman, and Ewertz.

> [T]he key distinction between the present invention and the closest prior art, is that the Misra et al., and Ginter et al. systems and the Ewertz et al. system run at the operating system level and BIOS level, respectively. More specifically, the closest prior art systems, singly or collectively, do not teach licensed programs running at the OS level interacting with a program verification structure stored in the BIOS to verify the program using the verification structure and having a user act on the program according to the verification. Further, it is well known to those of ordinary skill of the art that a computer BIOS is not setup to manage a software license verification structure. The present invention overcomes this difficulty by using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS.

(*E.g.*, *id.* at ANCC000162.) In short, neither Ewertz nor the combination of Misra, Goldman, and Ewertz teaches licensed programs running at the OS level interacting with a program verification structure stored in the BIOS. (Ex. 2011 at ANCC000161–62.)

VIII. Reexamination of the ’941 Patent

107. I understand the ’941 patent was reexamined by the USPTO in Ex Parte Reexamination No. 90/010,560, ordered after Microsoft filed a request for reexamination. (Ex. 1001 at 9 (ex parte reexamination certificate); Ex. 2001 at 4.)

The Examiner found claims 1–19 patentable without amendment. (Ex. 1001 at 9.) Dr. Wolfe does not consider the ’560 reexamination file history or the references cited in that proceeding. Microsoft’s request for reexamination relied on two references:

- U.S. Patent No. 6,153,835 by Schwartz et al. (hereinafter “Schwartz”)
- U.S. Patent No. 5,734,819 by Lewis et al. (hereinafter “Lewis”)

I discuss these references below and summarize relevant portions of the examiner’s actions.

A. Schwartz

108. Schwartz discloses “an electronic scale system and method which is particularly suitable for mailing or shipping use.” (Ex. 2021 [Schwartz] at 1:25–27.) The disclosed scale system is designed to allow updating of the postage rates and other shipping charges without necessarily replacing the memory inside the scale system, while also preventing or deterring “unauthorized copying of software.” (*Id.* at 2:38–49.) The components of the device are shown in Figure 8.

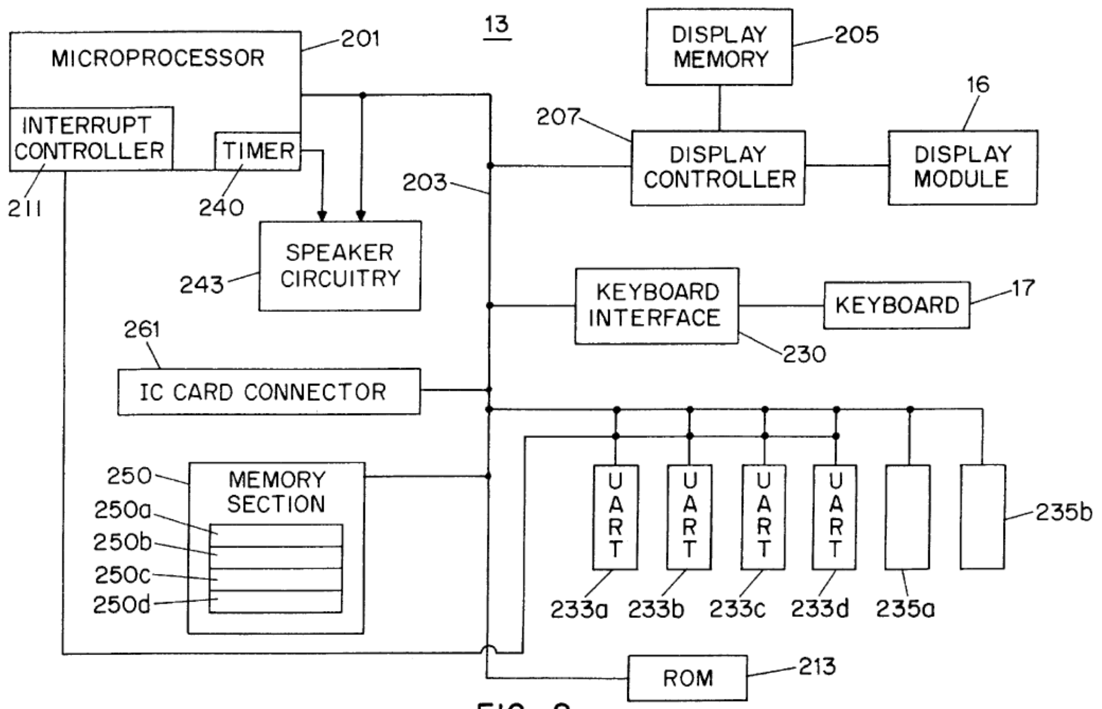


FIG. 8

(*Id.* at FIG. 8.) Notably, the memory section 250 includes a 128 Kbyte flash electrically erasable programmable read-only memory (EEPROM) 250a, a 32 Kbyte EPROM 250b, a 128 Kbyte nonvolatile static random-access-memory (SRAM) 250c, and a 128 Kbyte SRAM 250d. (*Id.* at 7:50–57.) The person skilled in the art would recognize that the EEPROM 250a and the nonvolatile SRAM 250c might be writeable in the context of Schartz's device. In contrast, the person skilled in the art would have recognized that the EPROM 250b would not be writable without removing it from Schwartz's system. Further, the person skilled in the art would have recognized that SRAM 250d would be writable, but volatile such that data would not be retained when power is removed from the device. Schwartz further illustrates the contents of the respective memories, in Figure 9.

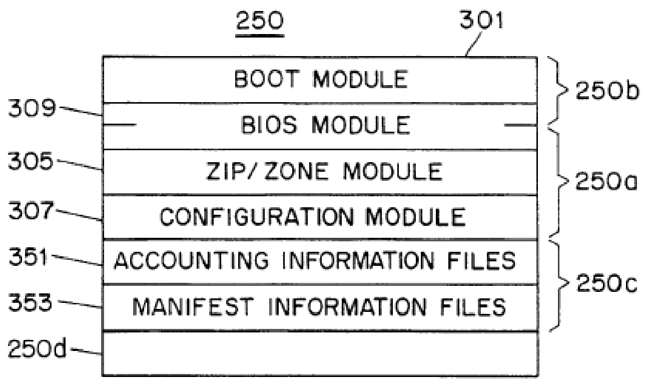


FIG. 9

(*Id.* at FIG. 9.) Schwartz explains that "part of BIOS module 309 [is] located in flash EEPROM 250a" and the "remaining part of BIOS module 309 and boot module 301 are located in EPROM 250b." (*Id.* at 8:17–20.)

109. As shown in FIG. 8, the disclosed scale system also includes an integrated circuit (IC) card connector 261. (*Id.* at 7:40–47.) Schwartz explains that rate schedule data, an operating system and an application program are provided in an IC card. (*Id.* at 8:26–31.) In one embodiment, Schwartz discloses executing a program off the IC card 401, and using rate schedule data in the IC card as needed. (*Id.* at 8:60–67.) In an alternative embodiment, Schwartz discloses that "memory section 250 in console 13 is supplemented with a flash EEPROM having a 1Mbyte capacity," and further that "rate schedule, operating system and carrier service program are accommodated in and run off the flash EEPROM, as opposed to IC card 401." (*Id.* at 9:1–5.)

110. In this alternative embodiment, Schwartz discloses aspects related to loading rate schedule data from the IC card 401 to the flash EEPROM (*Id.* at 9:34–65) and aspects related to loading new application code to the flash EEPROM (10:15–20). For example, the user needs to enter a valid authorization number to enable the new application software, or other new data, or system options selected by the user. (*Id.* at 10:21–25.) The authorization number is compared against a generated signature comprising (a) the serial number of the system 10, (b) the model number of system 10, (c) the version number of the application software, (d) the version number of the rate schedule data, (e) the version number of the zip/zone data, and (f) a 32-bit option number. (*Id.* at 10:25–49, 11:58–12:14, FIG. 12.) If the authorization number matches the signature, the authorization number is stored in configuration module 307. (*Id.* at 10:49–54.) The system performs a similar verification each time it is powered up, comparing the authorization number stored in configuration module 307 with a generated signature. (*Id.* at 11:24–40.)

111. Among other information, the generated signature includes the model number of the system, which is stored in BIOS module 309. (*Id.* at 11:27–28.) The model number, however, is unchanging and only identifies the model number (presumably meaning the hardware) of the electronic scale system. The remaining values, including the serial number of the system 10, the version number of the application software, the version number of the rate schedule data, the version

number of the zip/zone data, and the 32-bit option number; are stored in modules other than the BIOS module 309. (*Id.* at 11:23–33.) Likewise, the authorization number is stored in the configuration module 307, which is also separate from the BIOS module. (*See id.* at 10:49–54.)

112. The examiner found that the grounds based on Schwartz did not present a substantial new question of patentability, and did not order reexamination based on Schwartz:

> Schwartz discloses a postage scale that may receive new programs and store licensing information in related to these programs in non-volatile memory. See figure 9. The programs that provide this functionality, however, do not reside in BIOS; rather, they are instantiated as applications running on the operating system. It is therefore the case that the table created cannot be considered to be in BIOS either. ***Schwartz is therefore merely cumulative to the art cited by the Examiner during prosecution***, insofar as it teaches to the claim limitations.

(Ex. 2001 at 165 (emphasis added).) In short, the examiner seems to have recognized that (a) Schwartz does not disclose the claimed verification structure set up in the erasable, non-volatile memory of the BIOS, which includes at least one license record; and (b) Schwartz does not disclose the claimed agent used to set up the verification structure therein.

B. Lewis

113. Lewis discloses "a computer system having a non-volatile memory and a way of detecting when that information has been altered so as to prevent operation of the computer system once tampering has been detected." (Ex. 2020 [Lewis] at 1:11–15.) Lewis explains that electronic serial numbers are used, for example, to provide "a unique machine identification so that a software key is required to run on that specific machine serial number (this feature is provided in ***license managers*** such as NETLS)." (*Id.* at 1:18–23 (emphasis added).) Lewis describes a problem relating to this use of electronic serial numbers. "Since the serial number is ***located in a programmable memory***, it is easy for someone else to duplicate the serial number by simply copying the contents of one NVM media to another NVM media or writing a portion of the NVM media." (*Id.* at 1:37–41 (emphasis added).)

114. Lewis discloses a computer in which a device 16 includes a chip ID register 18, which includes a unique chip identifier within the chip ID register 18. (*Id.* at 4:33–34.) The computer further includes a non-volatile memory (NVM) 20 that stores the chip ID, along with other information about the device such as the DEVICE TYPE, DEVICE SERIAL NUMBER, and any other UNIQUE DEVICE DATA. (*Id.* at 4:40–44.) The non-volatile memory is illustrated in Figure 2:

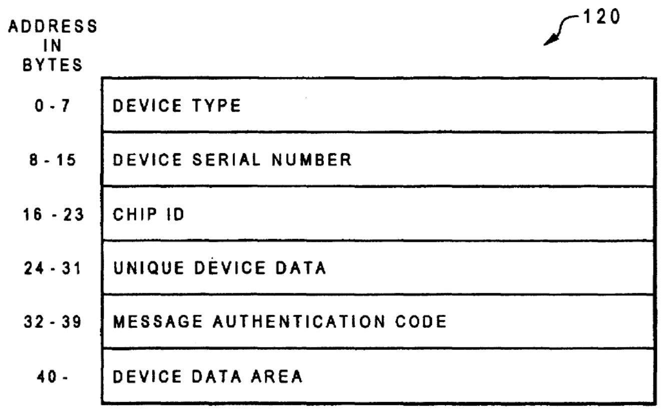


Fig. 2

(*Id.* at FIG. 2.)

115. Lewis further discloses a process for generating the message authentication code (MAC) shown in Figure 2 and storing the MAC in the non-volatile memory. (*Id.* at 4:55–5:9.)

116. Lewis also discloses a two-level process for verifying a computer system. “Prior to using device 16, the system code performs a chip identification and NVM content alteration detection test.” (*Id.* at 5:28–30.) First the system uses the stored MAC to provide the NVM content alternation detection test, in which the system generates a Message Authentication Code, once again using the first 32 bytes of the NVM data shown in Figure 2, and compares the generated value to the MAC stored in bytes 32–39 of the NVM. (*Id.* at 5:31–40.) If the MAC values match, “the system compares the chip ID field from bytes 16–23 of the NVM data stored in memory 12 with the chip ID field read from chip ID register 18.” (*Id.* at 5:40–50.) If

either verification comparison fails, the device 16 cannot be used. (*Id.* at 5:38–40, 5:45–50.)

117. In the paper ordering reexamination, the examiner stated that:

> Lewis discloses the loading into system memory (volatile) of a program, for which an encryption code (a MAC) is constructed using a driver for an external device in nonvolatile RAM. ***It is common in the art to implement such drivers in the BIOS area.*** The driver is used to write the MAC, which is derived using the computer's chip ID, to a table in non-volatile RAM, in order to use it later to verify that the program is on the computer on which it was installed. The correlating of specific instantiations of programs to specific computers constitutes a ***de facto license*** for that computer to use the program. Since the art cited during prosecution did not show such information being stored in and used from the memory of the BIOS, it is agreed that a reasonable examiner would have found this reference important in determining the patentability of claims 1-19.

(Ex. 2001 at 164–65 (emphasis added).)

118. Without issuing any rejections, the examiner concluded that Lewis did not anticipate or render claims 1–19 of the ’941 patent obvious. According to the examiner,

> Lewis discloses an invention that stores license information in non-volatile memory (which is the BIOS, since it is being setup and used by the system program) related to a system device, such as a DASD device, tape reader or diskette reader, or a cache controller, for which a program

> having instructions to control that device (a device driver) is instantiated in volatile memory (see Lewis, column 4, lines 25-31). Although the program is clearly associated with the device, the verification structure that is set up in non-volatile memory by Lewis is derived from a combination of non-functional descriptive material and information on the device itself, rather from the substance of the device driver, and is ***only being used to verify the device itself*** (or the information for the device written to non-volatile memory) and not the program that drives the device. Lewis' invention is not being used to verify the program (as per claim 1) or for verifying the application software program (as per claim 18), but rather ***just the device*** that the program is being used to access (see Lewis, column 5, lines 27-49).

(*Id.* at 249–50 (emphasis added).)

119. As described above, Lewis discloses a solution to problems stemming from the fact that serial number data is stored in "a programmable memory" that is non-volatile. (*See* Ex. 2020 [Lewis] at 1:37–41.) Lewis further discloses writing an encryption code (a MAC) into the nonvolatile memory. (*See id.* at 4:55–5:9.) The examiner reasoned that the driver writing the MAC to the non-volatile memory was part of the BIOS, because "[i]t is common in the art to implement such drivers in the BIOS area." (*See* Ex. 2001 at 164.) Further, the examiner ordered reexamination because "art cited during prosecution did not show such information being ***stored in and used from the memory of the BIOS***." (*Id.* at 164 (emphasis added).) Nevertheless, the examiner found claims 1–19 valid over the art cited in the

reexamination and in the prosecution history.

IX. Claim Construction

120. I have considered the claims in view of the claim construction standards described above in paragraphs 37–42.

A. "Agent"

121. Dr. Wolfe does not discuss the meaning of the term "agent" in his declaration. As discussed further below, I believe Dr. Wolfe's failure to consider the context surrounding the term "agent" leads him to incorrectly evaluate the validity of the '941 patent in view of the references identified in his declaration. As discussed below, the term "agent" carries specific meaning in view of its ordinary technical meaning and the prosecution history.

122. Independent claim 1 of the '941 patent recites:

> using an ***agent*** to set up a verification structure in the erasable, non-volatile memory of the BIOS, the verification structure accommodating data that includes at least one license record.

(Ex. 1001 at 6:64–67.)

123. Dependent claim 7 recites

> wherein using an ***agent*** to set up the verification structure includes the steps of: establishing or certifying the existence of a pseudo-unique key in a first non-volatile memory area of the computer; and establishing at least one license-record location in the first nonvolatile memory area or in the erasable, non-volatile memory area of the BIOS.

(*Id.* at 7:39–45.)

124. Dependent claim 14 recites:

> wherein the step of using the ***agent*** to set up the verification record, including the license record, includes encrypting a license record data in the program using at least the unique key.

(*Id.* at 8:8–11.)

125. Similarly, independent claim 18 recites:

> using an ***agent*** to perform the following steps:
>
> > extracting license information from software program;
> >
> > encrypting license information using the pseudo-unique key stored in the first non-volatile memory area;
> >
> > storing the encrypting license information in a second erasable, writable, non-volatile memory area of the BIOS of the computer;
> >
> > subsequently verifying the application software program based on the encrypted license information stored in the second erasable, writable, non-volatile memory area of the BIOS; and
> >
> > acting on the application software program based on the verification.

(*Id.* at 8:39–52.)

126. As a first matter, the person skilled in the art would have been familiar with the term "agent" as a software program or routine, as shown by various publications existing at the time of the invention. The Telecommunications

Handbook (1999) states that "[a]n agent is a program, which … performs tasks on behalf of a user or an application." (Ex. 2007 at 2–11.) Similarly, the Microsoft Press Computer User's Dictionary (1998) states that an "agent" is "[a] program that performs a background task for a user [] when the task is done or some expected event has taken place." (Ex. 2008 at 13.) Other references demonstrate the same understanding. (Ex.2009, Microsoft Computer Dictionary (4th Ed. 1999) at 18–19 (same); Ex. 2010, PC Magazine at 1 (An agent is "a software routine that waits in the background and performs an action when a specified event occurs.).)

127. In the *Ancora v. LG Electronics* case, the Western District of Texas interpreted "agent" as having a plain and ordinary meaning that requires "a software program or routine." (Ex. 1012 at 3 [LG CASE], Ex. 1013 at 28–36, 37.) I understand that the PTAB is not obligated to follow these decisions, but I recognize them as further evidence of the correct baseline technical definition of "agent."

128. The claims recite "agent" in several examples, summarized above, but the term "agent" does not otherwise in the '941 patent. The specification describes the agent's prescribed actions of "set[ting] up the verification structure" as "using E^2PROM manipulation commands" to store, add and modify items in the "erasable, non-volatile memory of the BIOS." (Ex. 1001 at 1:65–2:9.) The '941 patent further describes setting up the verification structure in the preferred embodiment:

> Setting up the verification structure includes the steps of: establishing or certifying the existence of a pseudo unique key in the first non-volatile memory area; and establishing at least one license-record location in the first or the second nonvolatile memory area.
>
> Establishing a license-record includes the steps of: forming a license-record by encrypting of the contents used to form a license-record with other predetermined data contents, using the key; and establishing the encrypted license-record in one of the at least one established license record locations (e.g. 10-12 in FIG. 1).

(*Id.* at 6:18–28.) The person having ordinary skill would therefore recognize that encompassing the preferred embodiment would require the claimed agent to be capable of accessing the first non-volatile memory area, and further to be capable of accessing the second nonvolatile memory area. (*See id.*) The '941 patent discloses the first non-volatile memory area as a "ROM section of the BIOS" and the second non-volatile memory as a "EEPROM section of the BIOS." (*See id.* at 5:12–16, FIG. 1.) Further, the person having ordinary skill would recognize that encompassing the preferred embodiment would require accessing the contents used to form the license-record, i.e., accessing the licensed software program. (*See id.* at 6:7–10.)

129. As discussed above in paragraphs 83–84, 96, 104–106, the term "agent" was added during prosecution to emphasize that "the closest prior art systems, singly or collectively, do not teach licensed programs running at the OS level interacting

with a program verification structure stored in BIOS." (*See* Ex. 2011 at ANCC000162.) As discussed above, applicants argued that Ewertz discloses storing identification information such as software licenses numbers "in a non-writeable, non-erasable area of the BIOS during manufacture." (*Id.* at ANCC000152.) Applicants also distinguished the claims from Ewertz and Misra, by arguing that Ewertz disclosed BIOS routines writing to the BIOS memory area while Misra disclosed an OS program. (*See id.* at ANCC000153–54.) Applicants argued that the respective OS program and BIOS program "cannot run at the same time." (*See id.*) Similarly, applicants argued that the invention "proceeds against conventional wisdom in the art," because "[t]he BIOS area is not considered a storage area for computer applications." (*Id.* at ANCC000153–54.) Consistent with these statements, a person of ordinary skill in the art would recognize that the claimed agent must run at the OS level and be capable of writing to memory of the BIOS.

130. The examiner ultimately agreed with applicants that the cited references did not teach licensed programs running at the OS level interacting with a program verification structure stored in the BIOS. (Ex. 2011 at ANCC000161–62.) Reviewing the examiner's statement, a person of ordinary skill in the art would recognize that the claims were ultimately allowed because the claimed agent must run at the OS level and be capable of writing to memory of the BIOS.

131. In view of both the applicant's and the examiner's statements, the term

"agent" in the context of the '941 patent would require an OS-level software program or routine. The claimed "agent" would be understood as a software program or routine separate from the BIOS.

132. I understand that the Federal Circuit decided several issues related to the '941 patent, and has similarly explained that "the applicants distinguished their invention over a combination of two references: one disclosed storage in the BIOS memory area by the BIOS software itself; the other disclosed software implemented in or through an operating system. The applicants explained that their invention differed from the prior art in that it both operated as an application running through an operating system and used the BIOS level for data storage and retrieval—a combination that was not previously taught and that an ordinarily skilled application writer would not employ." *Ancora Techs., Inc. v. Apple, Inc.*, 744 F.3d 732, 735-36 (Fed Cir. 2014).

133. I further understand that at least one district court has described the "agent" as being software. (Ex. 1012 at 3; Ex. 1013 at 28–37.) The Court also stated that:

> the [prosecution] history clearly recites an agent as a 'licensed program[] running at the OS level interacting with a program verification structure stored in BIOS.' *See* ECF No. 44, Ex. 2 at 7. The patentees emphasized this point during prosecution that agent was added to overcome the prior art, not to disclose additional hardware.

> *See* ECF No. 50 at 8. The Examiner also shared this understanding when referring to agent. *See* ECF No. 44, Ex. 9 at 4. A person of ordinary skill in the art would understand that agent is referring to software because of its complete interaction with the OS. *See* '941 Patent at 6:18-28; ECF No. 44, Ex. 2 at 7.

(Ex. 1013 at 34; *see also* Ex. 1002 at 33.)

X. The References Addressed in the Wolfe Declaration

134. In his declaration, Dr. Wolfe addressed the following references in making his assertions regarding the '941 patent:

- U.S. Patent No. 4,658,093 to Hellman ("Hellman");
- U.S. Patent No. 5,892,906 to Chou, et al. ("Chou"); and
- U.S. Patent 5,933,498 ("Schneck").

A. Hellman

135. Hellman discloses a "secure software distribution system in which the number of uses of software can be controlled." (Ex. 1004 at 1:5–7.) The system is designed to solve a "software piracy" problem, and further to allow software to be sold on a per use basis. (*Id.* at 1:8–23, 4:21–27, 4:34–36.) To accomplish this, Hellman discloses a "base unit" 12 consisting of particular hardware and designed to operate within a pay per use software control system. (*Id.* at 5:39–56.)

136. As shown in Hellman's Figure 1, the base unit 12 is separate from the authorization and billing unit 13. (*Id.* at FIG. 1, 5:39–46.) "Communication is

effected over the insecure channel 11 between the base unit 12 and the authorization and billing unit 13 using transmitter receiver units 14 and 16." (*Id.* at 5:42–46.)

137. Hellman discloses a series of values used in the secure distribution system:

- "SOFTWARE NAME is the name of the software package to be used." (*Id.* at 5:62–63.)
- "SERIAL NUMBER is a serial number, identification number, user name or similar identifier unique to base unit 12." (*Id.* at 5:63–65.)
- "N is the number of additional uses of software requested." (*Id.* at 5:65–66.)
- "R is a random number, counter value, or other non-repeating number generated by the base unit 12." (*Id.* at 5:66–68.) "The random number R varies from request to request, so that replay of an old authorization will not be accepted as valid unless the two R values are the same." (*Id.* at 9:58–60, *see also id.* at 9:60–63.)
- "BILLING INFORMATION is a credit [card] number or similar means for billing the user for use of the software." (*Id.* at 5:68–7:2.)
- SK is a base unit's secret key. In Hellman's symmetric key variant, SK is stored at the authorization and billing unit 13 in a table of serial numbers and secret keys. (*Id.* at 6:17–21.) In its public key variant, "The table of serial numbers and secret keys in memory 19, in FIG. 2, can then be eliminated. One secret key value, SK, would suffice for all base units". (*Id.* at 11:32-35.)
- K is a base unit key, stored in the base unit's permanent memory 31, for example a PROM which is burned in during manufacture of the base

unit. (*Id.* at 9:29–32.) Hellman discloses at least one embodiment where the base unit key K is the same as the secret key SK (*id.* at 9:32–40), and a different embodiment utilizing a public key cryptosystem such that K and SK are different (*id.* at 11:20–41).

- "H is used as an 'abbreviation' or name for describing the software package 21." (*Id.* at 6:33–35.) Hellman includes extensive discussion of "one-way hash functions" that are used to create this value and others. (*Id.* at 3:20–56, 7:17–8:12.)
- Authorization A is a value calculated in cryptographic function generator 23, from H, R, N, and SK. (*Id.* at 6:62–7:2.)
- Check value C is compared against authorization A. "If each bit of C matches the corresponding bit of A then the comparator 39 and the cryptographic check unit 34 generate a signal which indicates that A is to be considered a proper authorization and that the update unit 36 is to add N authorized uses to the software package with hash value H. If even one bit of C differs from the corresponding bit of A then A is not considered to be a proper authorization." (*Id.* at 10:19–26.)
- M is "the number of uses of software package 17 which are still available." (*Id.* at 10:42–43.) This value is stored in the base unit's non-volatile memory 37, using H as an address. (*Id.* at 10:38–43.)

138. Hellman illustrates features of the base unit in three drawing figures, corresponding to three phases of its operation. (*Id.* at 8:52–57.) Hellman explains that "[o]nly those elements of the base unit 12 which are needed in a particular phase are shown in the corresponding figure." (*Id.* at 8:57–60.)

139. Figure 5 illustrates the base unit during generation of a request for software use. (*Id.* at 8:61–62.)

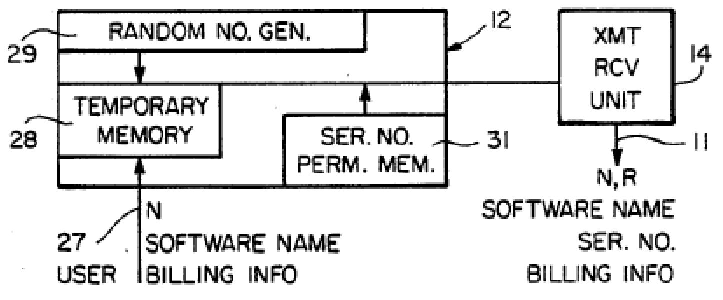


FIG_5

(*Id.* at FIG. 5.) "A user 27 communicates signals representing SOFTWARE NAME, BILLING INFORMATION and N, the number of additional uses desired, to the base unit 12, for example by typing them into a keyboard which is part of base unit." (*Id.* at 8:62–67.) Hellman discloses that the random number R is generated at the base unit. In the disclosed embodiment, random number generator 29 is "a noisy operational amplifier with hard quantization." (*Id.* at 9:2–7.) The person skilled in the art would recognize such a device as hardware. R is stored in temporary memory 28 for later use during verification of the received authorization. (*Id.* at 9:2–7.) The base unit's serial number is retrieved from a permanent memory 31, disclosed by Hellman as "a PROM which was burned in during manufacture of the base unit." (*Id.* at 9:8–10.) Such a PROM would be understood not to be writeable. (*See id.* at 9:32–40.) Hellman discloses that "[e]ncapsulating the base unit permanent memory

31 ***in epoxy or other means*** can be used to deter all but a very small percentage of users from learning their secret keys." (*Id.* at 9:46–49 (emphasis added).)

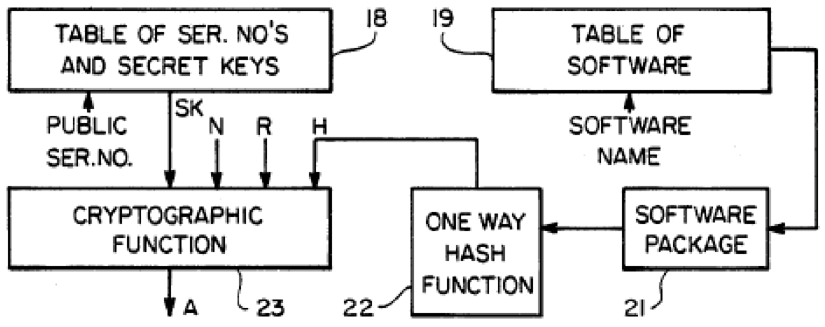


FIG_2

(*Id.* at FIG. 2.) At the authorization and billing unit 13, shown in Figure 2, the secret key SK is retrieved from "a table of serial numbers and secret keys" stored in a memory 18, based on the serial number provided by the base unit. (*Id.* at 6:17–21.) A "one-way hash generator 22" is used to produce H using a software package 21 retrieved from "another portion of the memory or in an additional memory 19" based on the SOFTWARE NAME provided by the base unit. (*Id.* at 6:21–30.) Cryptographic function generator 23 produces "a signal representing authorization A" from "four input signals" that include SK, N, R, and H. (*Id.* at 6:62–7:2.) In one embodiment, A is calculated as the output of a one-way hash function using H, R, and N as inputs. (*Id.* at 8:13–18.) Alternatively, A is calculated using a modified Data Encryption Standard (DES) using SK as the input to its key port and H, R, and N as inputs to the plaintext port. (*Id.* at 8:23–28.)

140. Figure 6 depicts the base unit during verification of an authorization A to use a software package. (*Id.* at 9:16–18.)

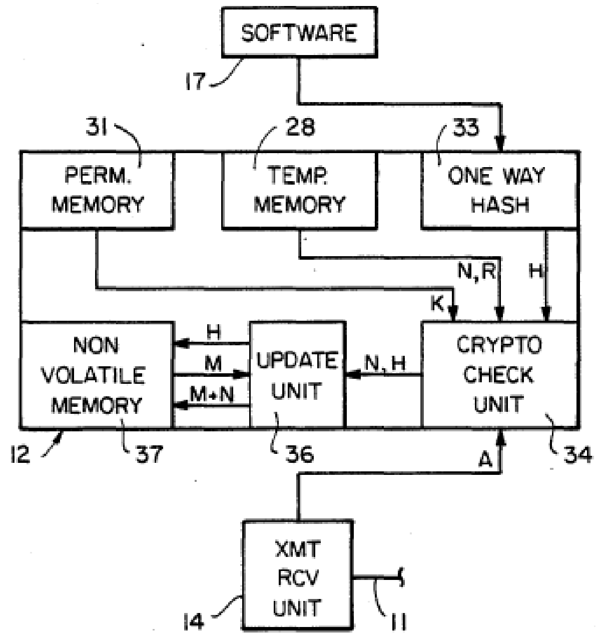


FIG_6

(*Id.* at FIG. 6.) The base unit determines whether authorization A is valid, using a cryptographic check unit 34 described in detail in Figure 7. If A is valid, the cryptographic check unit "applies signals representing N and H to an update unit 36." (*Id.* at 9:64–66.) The update unit 36 then increments the number of uses of software package 17 which are still available, stored in memory 37.

> Update unit 36 applies to interrogatory signal representing H to a non-volatile memory 37, for example an EEPROM or a CMOS memory with battery backup. The non-volatile memory 37 applies a signal to the update unit 36, said signal representing M, the number of authorized

uses of the software package H with hash value which still remain unused prior to this new authorization. The update unit 36 adds M and N and applies a signal representing M + N to the non volatile memory 37, so that M+N replaces the old number M in the non-volatile memory 37 as the number of uses of the software package which have been paid for."

(*Id.* at 10:1–13.)

141. Figure 8 illustrates the base unit during use of a software package. (*Id.* at 10:33–34.)

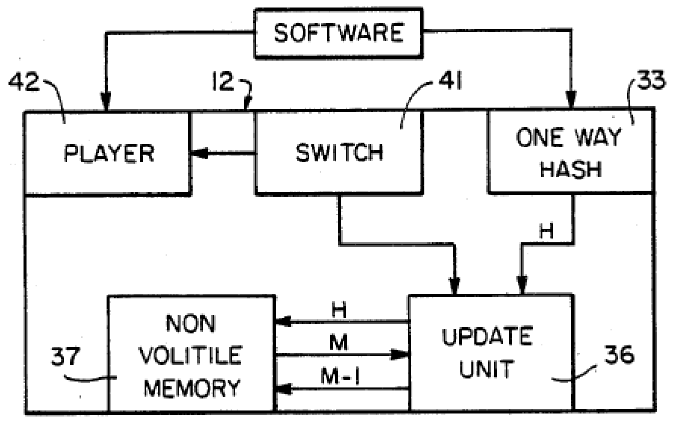


FIG_8

(*Id.* at FIG. 8.) In this phase, the base unit controls whether a player 42 can access software based on the value M stored in memory 37. In one embodiment, use of the software package 17 is allowed as long as M is greater than zero. (*Id.* at 10:44–54.) Alternatively, one value of M can represent an infinite number of uses. (*Id.* at 10:55–

65.) In either case, if M indicates available uses the update unit 36 "sends a control signal to switch 41 which activates software player 42, allowing it to use software package 17." (*Id.* at 10:44–46.) If M does not indicate available uses, the update unit does not send the control signal. (*Id.* at 10:50–54.)

142. The update unit 36 receives signals and generate signals to perform its function.

> Software package 17 is connected to the base unit 12 and a ***signal*** representing said software package is operated on by the one-way hash function generator 33 to produce an ***output signal*** which represents the hash value H. The ***signal H*** is transmitted to update unit 36 to indicate which software package is being used. Update unit 36 uses H as an ***address to non-volatile memory 37***, which responds with a ***signal*** representing M, the number of uses of software package 17 which are still available.

(Ex. 1004 [Hellman] at 10:33–43 (emphasis added).) The player 42 could be a variety of hardware devices. "For example, if the software is recorded music then software player 42 would be a record player; if the software is a computer program, then software player 42 would be a ***microprocessor or central processing unit (CPU)***." (*Id.* at 10:66–11:3 (emphasis added).)

143. Hellman does not disclose any BIOS or any operating system ("OS") or OS-level software. Dr. Wolfe also writes "Hellman does not disclose that the non-volatile memory 37 is a BIOS, or that the base unit 12 included BIOS." (Ex. 1003 at

¶ 105.)

B. Chou

144. Chou discloses an apparatus and method for discouraging computer theft. (Ex. 1005 [Chou] at Abstract.) This includes use of "a password or other unique information be supplied to the computer ***before the computer BIOS routines can be completely executed***." (*Id.* (emphasis added).) A person having ordinary skill in the art would recognize that Chou does not disclose any means of preventing software piracy. Chou's sole embodiment discloses a computer "which includes a security function stored as a programming routine within the BIOS EEPROM 15," along with other components illustrated below in an annotated copy of Figure 1.

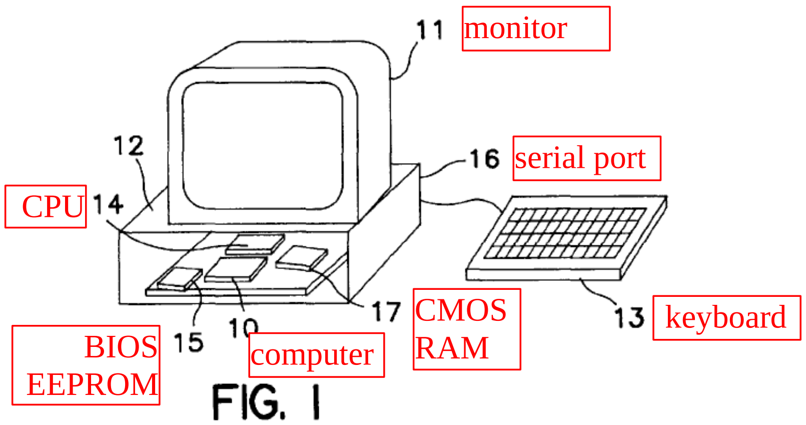


FIG. 1

(*Id.* at FIG. 1 (annotations added), 3:30–36.) Chou discloses that the BIOS memory 15 includes routines for exercising a security function 25. According to Chou, "[u]nless the BIOS routine has completely executed, the computer operating system can never be accessed rendering the computer inoperative." (*Id.* at 3:60–62;

see also id. at 4:1–5.)

145. Chou discloses two embodiments: (1) a dongle-based locking function (*id.* at 5:21–6:19), and (2) a password-based locking function (*id.* at 7:14-17; 8:42–9:50). In the first (hardware-based) embodiment, the only data stored in BIOS EEPROM is computer serial number and public key—neither of which ever change. In the example that requires a user entered password, the user passwords are stored in a BIOS memory.

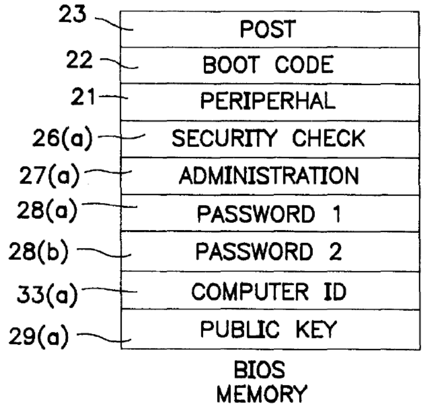


FIG. 7

(*Id.* at FIG. 7, 7:14–28.) In both embodiments, the computer's CMOS RAM includes a location that indicates whether the computer is in a locked state. (*Id.* at 7:31–44.)

146. Chou discloses "execution of the ***BIOS routines*** including the security function" in Figure 10. (*Id.* at 8:42–43 (emphasis added).)

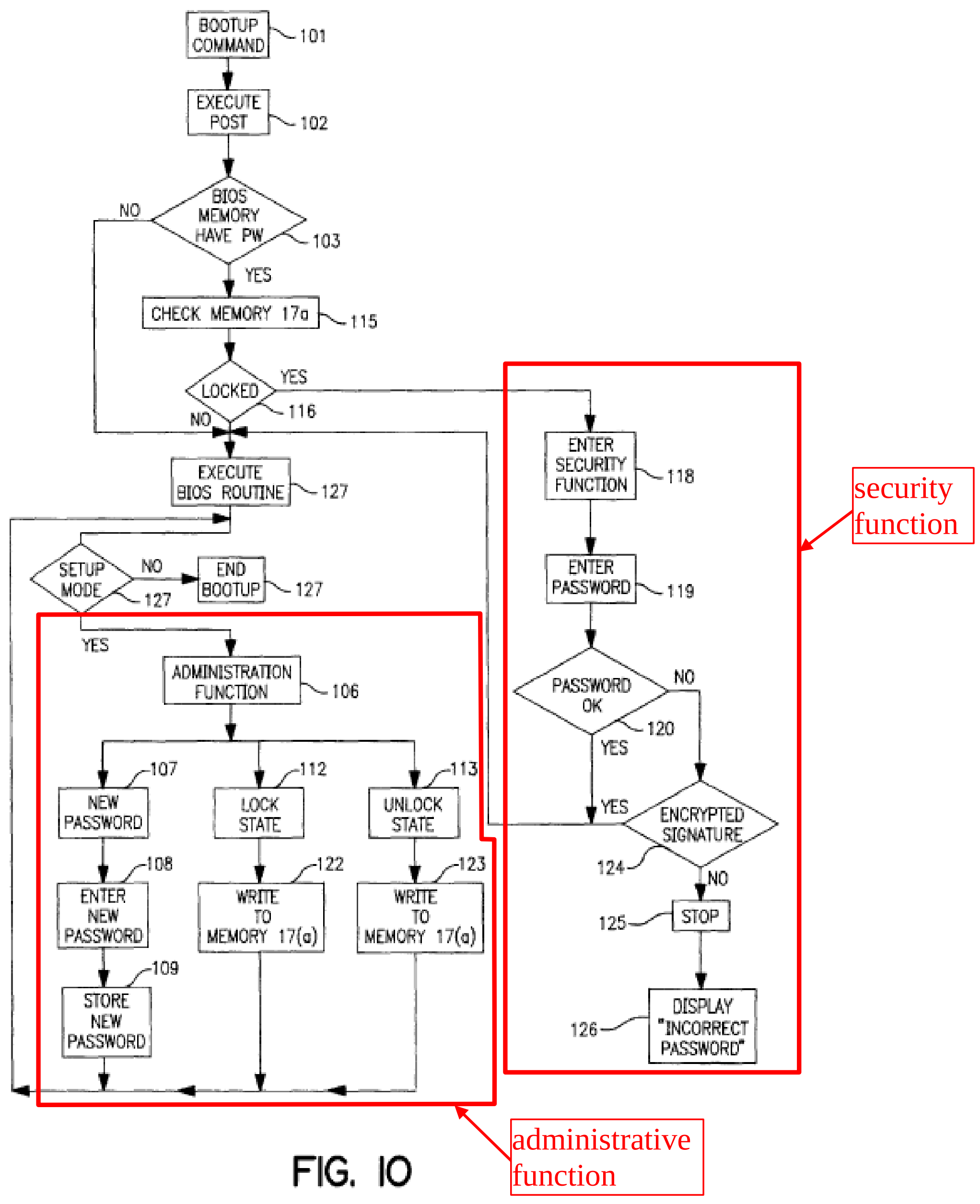


FIG. 10

(*Id.* at FIG. 10 (annotations added).) As called out above, Chou's security function 25 includes steps 118–126 requiring the user to enter a password if the computer is in a locked state. (*Id.* at 9:13–17, 9:26–36.) The system also includes administrative functions, called out above, that allow the user to enter a new password or to place the computer in the locked or unlocked state. (*Id.* at 8:49–61, 9:1–7.) Both the security function and the administrative function are part of BIOS routines that run before the computer can run its operating system.[2] (*Id.* at 8:49–55.) A person of ordinary skill in the art would understand that Chou's administrative function runs outside of the operating system. Further, the only means disclosed by Chou for changing the user-entered password is through the administrative function described above, running outside of the computer's operating system. (*See id.* at FIG. 10.)

C. Schneck

147. Scheck discloses a system that "relates to control of distribution and access of digital property as well as the payment therefor." (Ex. 1006 at 1:10–11.) Schneck explains that "electronic information has opened new questions about copyright, ownership, and responsibility for information." (*Id.* at 1:27–29.) Thus, "the threshold inhibiting the making of illicit copies is significantly lowered." (*Id.* at

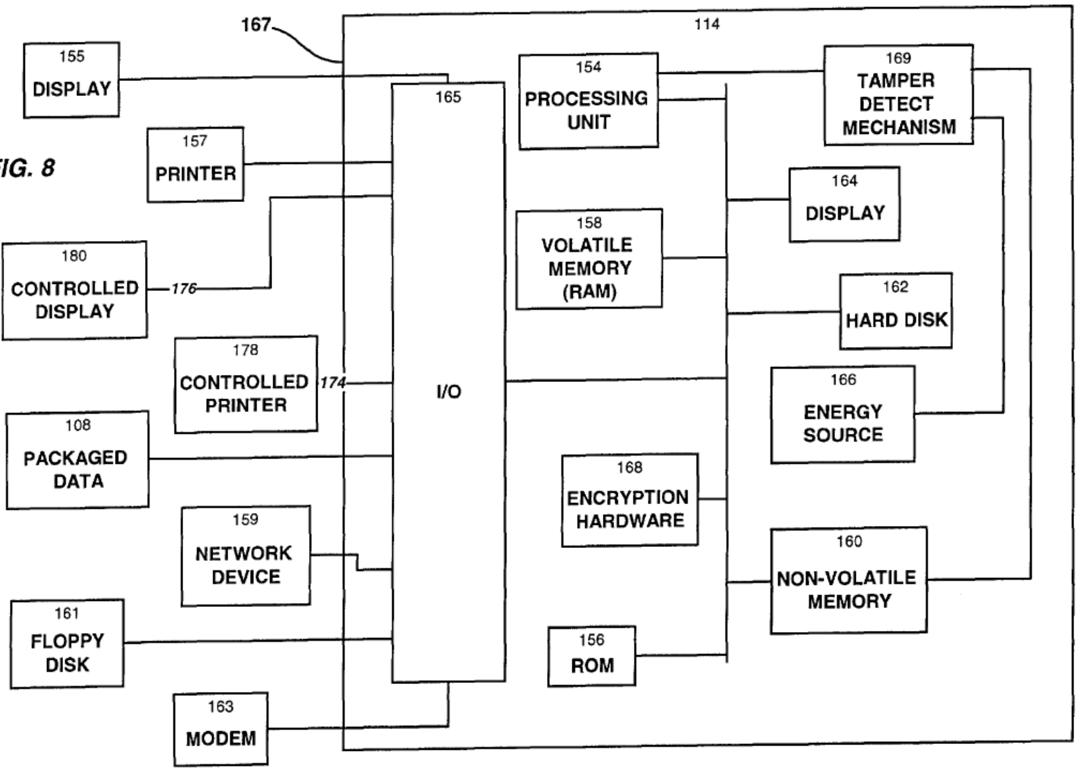
[2] I note that Chou's callouts on FIG. 10 do not match the text of the specification (e.g., two blocks are labeled 127 and none is labeled 104). The person skilled in the art, however, would understand the flow chart in conjunction with the text of the specification as I have described it.

2:51–52.)

148. Schneck disclose known means for protecting digital files, including "software-based cryptography" and "the use of external devices or tokens (dongles)." (*Id.* at 3:37–57.) According to Schneck, however, none of these systems protects data after it has been decrypted, allowing secondary distribution and multiple uses. (*Id.* at 3:58–60, 4:65–5:3.)

149. The invention disclosed by Schneck is designed to operate in a system comprising a distributor 102 (which uses an authoring mechanism 112 to produce packaged data 108) and a user 104 (which uses an access mechanism 114 to access data in controlled ways). (*Id.* at FIGS. 1 & 5, 9:46–59.) The access mechanism 114 is shown in greater detail in Schneck's Figure 8.

FIG. 8



(*Id.* at FIG. 8.) Schneck explains that:

> All components of the access mechanism 114 are ***packaged in such a way as to exclude any unknown access by a user*** and to discover any such attempt at user access to the components or their contents. That is, the access mechanism 114 is packaged in a tamper-detectable manner, and, once tampering is detected, the access mechanism is disabled. The line 167 depicted in FIG. 8 defines a so-called security boundary for the components of the access mechanism 114. Any components required for tamper detection (tamper detect mechanism 169) are also included as part of the access mechanism 114.

(*Id.* at 15:51–61 (emphasis added).) This use of physical self-protection measures is used as a means for rendering the access mechanism inoperative, among other passive and active mechanisms employed to destroy data. (*Id.* at 15:64–16:15.) Schneck explains that tamper detection allows the access mechanism 114 to destroy data before any tamperer can obtain the data. (*Id.* at 16:16–19.) Schneck describes a similar tamper detection and data destruction scheme even for an alternative embodiment in which the access mechanism is a co-processor of another processor or computer. (*Id.* at 16:49–59.)

XI. General Comments on the Wolfe Declaration

A. Frequent Use of Impermissible Hindsight

150. Having reviewed the assertions of obviousness made in the Wolfe Declaration, it is my opinion that the assertions of obviousness suffer from the evident use of impermissible hindsight. In short, Dr. Wolfe's report shows that someone appears to have looked at the limitations of the Challenged Claims and then used them to search for unrelated disclosures in order to patch together a combination of references that, when combined, allegedly render obvious the Challenged Claims of the '941 patent.

151. Aside from offering conclusory statements that a POSITA would have found it obvious to combine such references, Dr. Wolfe offers very little to explain how any of the proposed combinations of references would have been able to function—either in theory or, more importantly, in practice. Dr. Wolfe does not

explain how any of those proposed combinations actually could be implemented, identify any possible complications or further design considerations that must be met, or claim what likelihood of success would have been expected had a POSITA attempted to implement any of the proposed combinations. Dr. Wolfe also provides very little detail around the motivation that any person would have had to combine references.

B. Inadequacy of Addressing the Expectation of Success

152. As I noted above, it is my understanding that—when asserting invalidity based on a combination of references that allegedly disclose a given patent claim—such assertions must include a discussion of the likelihood that the proposed combinations can be made by those of ordinary skill and, if made, will have a likelihood of success.

153. I address some likely outcomes of the attempted combinations below. But even if one simply assumes that Dr. Wolfe's proposed combinations could be technically achieved, the results do not demonstrate an OS role in "using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS" under the construction provided by the Court, which recognized that the claimed agent operates at the operating-system level as explained in ¶133 above. Rather, the proposed combinations only attempt to show that a BIOS component can be made to store certain information in BIOS memory as opposed to the memory identified

in Dr. Wolfe's analysis of Hellman, Chou, and Schneck. The references show no OS-to-BIOS interaction, no such interaction for purposes of storing data in the BIOS, and no such interaction for purposes of storing, specifically, a license record in the BIOS. Nor do the references even acknowledge, as the '941 patent does, one of the benefits of using BIOS for this purpose: that "that the required level of system programming expertise that is necessary to intercept or modify commands, interacting with the BIOS, is substantially higher than those needed for tampering with data residing in volatile memory such as hard disk." (Ex. 1001 ['941 patent] at 3:5-9.) Ultimately, Dr. Wolfe's claimed combinations do not result in a system that meets the requirements of the '941 patent.

XII. Rebuttal to Dr. Wolfe's Opinions Regarding the '941 Patent

154. In this section of my declaration, I state and explain my opinion that Dr. Wolfe failed to show that the prior art addressed in his report invalidates any of the Challenged Claims of the '941 patent, and provide the bases for my opinions.

155. The Wolfe Declaration appears to agree that the '941 patent is entitled at least to a priority date of May 21, 1998. *See* Ex. 1003, Wolfe Declaration ¶ 25.

156. I further note that, due to my understanding of the law, I am not offering an opinion regarding each cited reference or combination of references beyond responding to the opinions and evidence identified by Dr. Wolfe.

157. Finally, I note that I agree with the analysis of the '941 patent

applicants, the USPTO Examiner, and the Federal Circuit recited above at §VII-VIII.

A. The Combination of Hellman, Chou, and/or Schneck Does Not Invalidate the Asserted Claims of the '941 Patent

158. Beginning at ¶ 98, Dr. Wolfe opines that Hellman in combination with Chou and/or Schneck renders obvious each Challenged Claim of the '941 patent. I disagree. It is my opinion that Dr. Wolfe has not shown that the challenged combinations make obvious claim 1 (or the remaining Challenged Claims, which depend on claim 1), including for the reasons I explain above at § XI.

159. For reference, I note that Dr. Wolfe asserts that certain '941 Patent Claim 1 elements are found in the combination of Hellman, Chou, and/or Schneck as follows:

> 160. "**memory [area] of [a/the] BIOS**": Dr. Wolfe states "It is my opinion that Chou discloses the 'erasable, non-volatile memory area of a BIOS,' and that a POSA would have found it obvious to include the 'erasable, non-volatile memory area of a BIOS' in Hellman's system." (Ex. 1003 at ¶ 98.)
>
> 161. "**verification structure**": Dr. Wolfe states "Hellman discloses a 'verification structure' in the form of the memory structure of non-volatile memory 37 storing at least one value M at memory addresses defined by at least one hash value H." (*Id.* at ¶ 135.) Further, Dr. Wolfe states "In light of

this disclosure of Schneck, a POSA would have found it obvious to modify Hellman to store the number of authorized uses value M in encrypted form in non-volatile memory 37… In considering how to store M in encrypted form, one technique that a POSA would have found obvious would have been to store the authorization A in non-volatile memory 37 at memory address H.” (*Id.* at ¶¶ 146-147.)

162. “**license record**”: Dr. Wolfe states “Hellman discloses a ‘license record’ in the form of the number of authorized uses value M.” (*Id.* at ¶ 134.)

163. “**agent**”: Dr. Wolfe states “Hellman discloses an ‘agent’ in the form of update unit 36.” (*Id.* at ¶ 137.)

1. Using an Agent to Set Up a Verification Structure in the Erasable, Non-Volatile Memory of the BIOS

164. I disagree with many of Dr. Wolfe’s conclusions with respect to the “using an agent…” element. (*See* Ex. 1003 ¶¶ 133–150.) First, I note that the Hellman disclosure is expressly predicated on the use of additional hardware including wires, switches, and glue. (Ex. 1004 [Hellman] at 10:33–43.) To the extent the software player 42 is a “microprocessor or central processing unit (CPU),” Hellman does not disclose that the other components of the base unit would or could be implemented using the same processor. (*See id.* at 10:66–11:3.) I disagree with Dr. Wolfe to the extent he assumes that such a microprocessor or CPU present as

"software player 42" would also be used to implement the switch 41 or the update unit 36, for example. As indicated below, Hellman's architecture requires protecting its sensitive components such as its switch 41 and update unit 36 from attack. Dr. Wolfe does not explain how a single "software player" CPU would be able to simultaneously "play" arbitrary software obtained from third parties while preventing such arbitrary software from accessing Hellman's non-volatile memory 37 or executing code performing functions of the switch 41 or update unit 36.

165. Hellman discusses the physical construction of the base unit, including its wires, switches, and glue. "Base unit 12 should be physically constructed so that ***switch 41*** is not readily accessible to the user. Otherwise a user could cut the ***wire*** sending the ***control signal*** from update unit 36 to software player 42 and force the control signal to always activate software player 42.[3] The use of ***epoxy*** to encapsulate ***switch 41 and the wire connecting it to update unit 36*** would be one such approach. It may suffice only to detect if a base unit 12 has been tampered with if the user signs a license agreement which says he will not open the base unit. Detection of tampering is a simpler task and techniques are well known for accomplishing this

[3] A POSITA would understand that this sentence misidentifies the connection at issue. As the following sentence of Hellman indicates, the wire between switch 41 and base unit 36 is the one under discussion.

task. For example, many warranties are void if equipment is opened by the user. Manufacturers of such equipment use ***special factory seals*** or a ***dab of paint*** across a joint to tell if a piece of equipment has been opened." (Ex. 1004 [Hellman Patent] at 11:4–19 (emphasis added).)

166. Describing the procedure used to modify Hellman's non-volatile memory in response to a new authorization, Hellman describes its cryptographic check unit and its comparator and input and output signals: "FIG. 7 depicts an implementation of the cryptographic check unit 34. ***Signals*** representing K, N, R, and H are applied as inputs to a cryptographic function generator 38 which generates a check value C as an output signal. Signals C and A are input to a comparator 39. If each bit of C matches the corresponding bit of A then the comparator 39 and the cryptographic check unit 34 generate a ***signal*** which indicates that A is to be considered a proper authorization and that the update unit 36 is to add N authorized uses to the software package with hash value H." (Ex. 1004 [Hellman] at 10:14–24 (emphasis added).) "***Encapsulating the base unit permanent memory 31 in epoxy or other means*** can be used to deter all but a very small percentage of users from learning their secret keys." (*Id.* at 9:46-49 (emphasis added).)

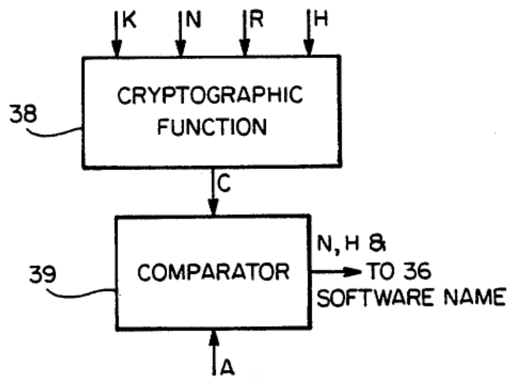


FIG_7

167. The output of the comparator shown above includes "N, H & Software name" and leads to Update Unit 36 shown in Figure 6.

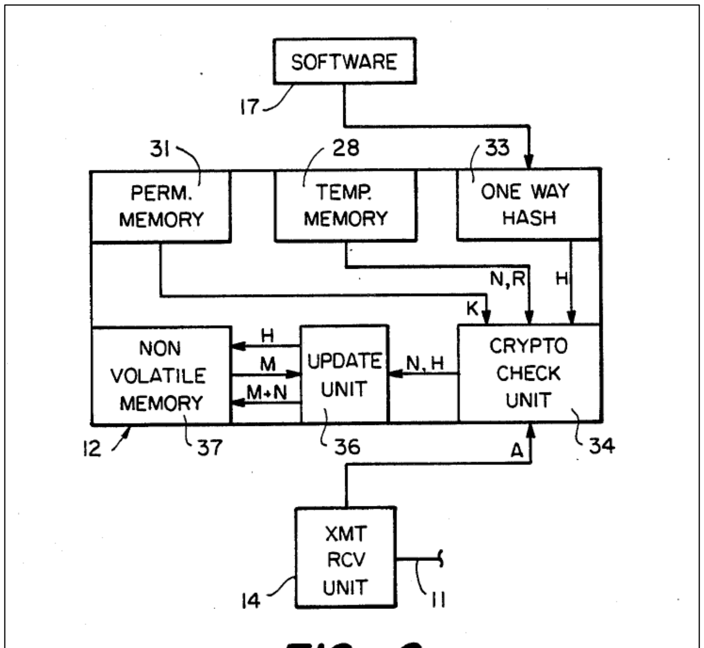


FIG_6

168. In fact, every use of the word "software" in Hellman refers to the software package being authorized for use a given number of times by a base unit. Therefore, Hellman does not disclose a "software program or routine" (agent) that is used to set up a verification structure. Dr. Wolfe admits this. He states at ¶ 137 that "Hellman does not specifically disclose how update unit 36 is implemented" and simply speculates that "a POSA would have recognized that the update unit 36 would have been implemented by a software routine, potentially along with a hardware module" and further that "the update unit 36 would have been implemented by software, hardware, or some combination of the two." (Ex. 1003 at ¶¶ 137–137A.)

169. Dr. Wolfe is incorrect. First, as I explained above, Hellman discloses that the "update unit 36" is a hardware unit that is connected by "wires" to "switches." Hellman also says that data can be "stored in" the update unit, further suggesting that it is not a "software program or routine," but rather something with memory: "The fixed value of N would only have to be stored in update unit 36." (Ex. 1004 [Hellman] at 12:34–36.)

170. In addition, Hellman teaches away from implementing its invention in software. Its abstract describes the base unit as a "computer, video game base unit, record player, videorecorder or video disk player," clearly indicating that the base unit is a hardware device. Corresponding to this, the "software" that Hellman may authorize for use a given number of times includes "records," e.g., LPs: "In the record industry, illegal home and commercial taping of records is depriving artists, recording studios, and manufacturers of significant income which is rightfully due them." (Ex. 1004 [Hellman] at 1:10–13.) "Software copy protection does not currently exist in the record industry." (Ex. 1004 [Hellman] at 1:28–29.)

171. Since a record player is a hardware device, not a general-purpose computer, this disclosure would not bring to mind a pure software implementation. Record players are not programmable and do not have "software" that can be adapted to Hellman's invention. Hellman's inclusion of videorecorders and video disk players in its Abstract reinforces this, as does Hellman's reference to "videotape"

and "disk" media as "software" (e.g., "Control of software is a major problem in the record, movie (videotape and disk), computer, and videogame industries. In the record industry, illegal home and commercial taping of records is depriving artists, recording studios, and manufacturers of significant income which is rightfully due them. A similar problem exists with illegal taping of movies in the videotape and videodisk industries." (Ex. 1004 [Hellman] at 1:8–15.)

172. Contrary to Dr. Wolfe's opinion about what Hellman "does not explicitly say," the person having ordinary skill in the art would have recognized many context clues indicating that Hellman intended the update unit 36 to be a hardware component. As noted above, Hellman discloses "signals," and "wires" connected to the update unit. Dr. Wolfe further testifies that the activities performed by the update unit 36 were "of a type that could be performed in software, hardware, or both. (Ex. 1003 at ¶ 137B.) But the only reason Dr. Wolfe offers for favoring a software-based update unit 36 is allowing the provider of the base unit to change the implementation logic of the update unit 36. As Dr. Wolfe recognizes elsewhere, implementing the update unit in software readily accessible to attackers would come with a set of security risks. For example, Dr. Wolfe contends that encryption of the value M stored in non-volatile memory 37 would have been desirable to resist attempts to "interrogate the non-volatile memory 37." (Ex. 1003 at ¶¶ 142–43.)

173. Dr. Wolfe's own assumptions are the source of his perceived problem

with the way Hellman's value M is stored. In Hellman's disclosure of hardware components, the values stored in non-volatile memory would be substantially protected against software-based means to "interrogate the non-volatile memory 37." Faced with the same problem, Schneck similarly relied upon ***hardware*** to protect unencrypted data. As discussed above, all components of Schneck's access mechanism are packaged in a tamper-detectible manner using physical means. (*See, e.g.*, Ex. 1006 at 15:51–16:19, 16:49–59.)

174. But even a POSITA who does happen to wonder whether the Hellman system could be adapted to pure software would immediately have to grapple with the question of how a software solution could isolate and protect itself from the very software packages whose authorization is at issue. Dr. Wolfe does not consider this issue. This is a complicated problem, and the universe of wildly different designs to address problems of software isolation indicate that the results of attempting it are highly unpredictable.

175. For example, operating systems that rely on well-established CPU memory protection features repeatedly succumb to software-based attacks. The Common Vulnerability Enumeration (CVE) database of Mitre Corporation lists over

1,500 publicly disclosed vulnerabilities for the year 1999 alone.[4] A computer system that attempts to isolate trusted software from untrusted software inevitably provides mechanisms to transition from untrusted modes to trusted modes and vice-versa. These mechanisms are themselves subject to attack by race conditions, buffer overflows, and other forms of "privilege escalation" attacks that allow rogue software greater access to the system's memory than intended. Even when a computer system architecture succeeds in preventing rogue software from accessing forbidden memory using ordinary access methods, the fact of running rogue software in close proximity to a protected element may reveal secret information through side-channel attacks. For example, rogue software that is able to observe detailed timing patterns of a CPU's total workload over time when that CPU is performing as-designed cryptographic operations may be able to deduce the cryptosystem's underlying secret keys.[5] To suggest that it is simply obvious to deploy Hellman as

[4] *See* https://www.cve.org/Downloads, https://cve.mitre.org/data/downloads/allitems.html, reviewed April 29, 2022.

[5] *See* Ex. 2046. Paul C. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*. In Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '96).

"software program or routine" *and* combine that highly underspecified system with Chou or Schneck to result in a "software program or routine" that operates at the OS-level to set up a verification structure in memory of the BIOS while maintaining the security properties of Hellman is highly implausible.

176. Therefore, Dr. Wolfe's alleged combinations of Hellman with Chou or Schneck must be understood as combinations of *a fundamentally transformed, software-implemented Hellman* with Chou or Schneck. Dr. Wolfe has neither established any reasonable motivation for this transformation, nor has he fully considered the reasons against such a change or even fully described the needed modifications.

177. Further, even if I were to agree that the person having ordinary skill in the art would have implemented Hellman's update unit 36 and/or authorization and billing unit 13 as software, Dr. Wolfe's analysis still fails to establish that any of these elements of the combined system would have been implemented at the OS-level. Dr. Wolfe fails to consider the limiting statements made by both the applicant and the examiner during prosecution of the '941 patent. As discussed above at

---

Springer-Verlag, 1996, pp. 104–113. This paper describes a timing attack against the "Diffie-Hellman" algorithm that was introduced in the same paper cited in Ex. 1004 [Hellman] at 3:29-44.

beginning at ¶ 121, the claimed "agent" would run at the OS level.

178. As stated above at ¶ 143, Hellman does not disclose an operating system or a BIOS. It therefore completely fails to disclose an agent that is separate from the BIOS but capable of setting up a verification structure in the erasable, non-volatile memory area of a BIOS—let alone an OS-level software program or routine.

179. Similarly, Chou explicitly discloses "BIOS routines" used to update the passwords stored in the computer BIOS, as discussed above at ¶ 146. Nothing in Chou teaches an agent that is separate from the BIOS but capable of setting up a verification structure in the erasable, non-volatile memory area of a BIOS—let alone an OS-level software program or routine.

180. Dr. Wolfe makes no attempt to show that his proposed combination would have incorporated an agent that is separate from the BIOS but capable of setting up a verification structure in the erasable, non-volatile memory area of a BIOS—let alone running as an OS-level software program or routine.

181. At ¶ 138A, Dr. Wolfe argues that Hellman's "authorization and billing unit 13 may be considered an agent." He also argues that "the authorization and billing unit 13 may ***cooperate with*** the update unit 36 to act as the 'agent.'" (Ex. 1003 at ¶ 138 (emphasis added).) However, he does not claim that Hellman's "authorization and billing unit 13" plays the role of "agent" in the claim 1 limitation of "using an agent to set up a verification structure in the erasable, non-volatile

memory of the BIOS". Dr. Wolfe's silence makes sense, because Hellman's authorization and billing unit 13 has no access to the base unit's non-volatile memory. Rather, Hellman teaches that "Communication is effected over the insecure channel 11 between the base unit 12 and the authorization and billing unit 13 using transmitter receiver units 14 and 16, which may be modems such as Bell 201 modems. Transmit-receive units 14 and 16 could be humans conversing over a telephone line. The human at the transmit-receive unit 16 would then type the voice information into a keyboard for entry in the unit 13." (Ex. 1004 at 5:42–50; *see also* Figures 1 and 6.)

2. Alleged Memory of the BIOS

182. Dr. Wolfe recognizes that Hellman is also silent on BIOS, writing at ¶105 regarding the preamble of claim 1 that "Hellman does not disclose that the non-volatile memory 37 is a BIOS, or that the base unit 12 included BIOS." Dr. Wolfe argues at ¶107 "A POSA would have found it obvious to include BIOS in the base unit 12 of Hellman."

183. In ¶ 109, Dr. Wolfe attempts to justify Hellman's silence on BIOS by writing that BIOS terminology "was not as consistently used for other types of electronic devices" such as a videogame bas unit or a videorecorder. However, a POSITA would not take Hellman's silence on BIOS to be an indication of a vacuum that must be filled. Rather, a POSITA would take Hellman's lack of BIOS as an

indication that BIOS and memory of the BIOS are simply irrelevant to Hellman's aims. Even if a POSITA did for some reason decide to combine Hellman with a BIOS stored in erasable, non-volatile memory, that combined system reveals nothing about "using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS" beyond what the Examiners of the '941 patent already considered when allowing the '941 claims.

184. This is reinforced by the fact that Hellman is very explicit about the types of memories it uses and the purposes those memories serve, but never discusses BIOS memory or suggests that it could or would serve any use or purpose for accomplishing Hellman's objectives. For instance, in addition to the non-volatile memory identified at Ex. 1004, Hellman at 10:1-5, Hellman discloses that "Authorization and billing unit 13 contains a memory 18 having a table of serial numbers and secret keys which allows authorization and billing unit 13 to determine a base unit's secret key, SK, from knowledge of the base unit's public serial number." Ex. 1004, Hellman at 6:16-21.

185. Hellmann also discloses "Authorization and billing unit 13 also contains in another portion of the memory or in an additional memory 19 a table of software which allows authorization and billing unit 13 to determine the complete contents of software package 17 from knowledge of the much smaller information SOFTWARE NAME." Ex. 1004, Hellman at 6:21-27. It also discloses the base unit

12's temporary memory as "Base unit 12 stores these values in a temporary memory 28, for example a RAM." Ex. 1004, Hellman at 8:66-67. And Hellman also discloses "permanent memory 31" as "a secure memory, inaccessible to the user," adding that "Encapsulating the base unit permanent memory 31 in epoxy or other means can be used to deter all but a very small percentage of users from learning their secret keys." Ex. 1004, Hellman at 9:30-47.

186. Hellman's omission of any discussion of BIOS or BIOS memory is thus more pronounced given Hellman's very specific description and identification of the range of memory types to be used in the invention as well as the purpose that each one serves. Nor does Hellman even suggest using the only "inaccessible" memory it discloses—permanent memory 31—as a location for the verification structure. Ex. 1004, Hellman at 9:29-49. Rather, Hellman discloses using such memory only for storing the "base unit key, K" to ensure that the key is "inaccessible to the user" and to prevent all but a "small percentage of users go to the trouble of learning their secret keys." Ex. 1004, Hellman at 9:35-44.

187. According to Dr. Wolfe in ¶ 136, however, Hellman teaches that update unit 36 to "sets up a structure… in the non-volatile memory 37" Notably, despite expressly disclosing the use of memory normally "inaccessible" to users to store "base unit key, K," Hellman never teaches or suggests using similarly inaccessible or protected memory to set up a verification structure. As I explain in ¶ 63 above,

and as the ’941 patent explains, “An important advantage in utilizing non-volatile memory such as that residing in the BIOS is that the required level of system programming expertise that is necessary to intercept or modify commands, interacting with the BIOS, is substantially higher than those needed for tampering with data residing in volatile memory such as hard disk.” Ex. 1001, ’941 patent at 3:4-9. Dr. Wolfe’s reference to “non-volatile memory 37” does not provide this property.

188. At ¶ 120, Dr. Wolfe argues that a POSITA combining Hellman with Chou would have decided to store Hellman’s authorization value M in memory inspired by Chou, which Dr. Wolfe characterizes as “memory of the BIOS.” But Dr. Wolfe cites no motivation, teaching, or suggestion to do either of these things, other than speculating that there may be EEPROM with varying memory space to permit it. And Dr. Wolfe does not opine on whether a POSITA would understand these operations to take place at the BIOS level or any other software level. There is certainly no reason to think that the asserted combinations would inevitably result in using OS-level software to set up a verification structure in a memory of the BIOS. Hellman mentions neither BIOS nor operating system.

189. Hellman discloses that its non-volatile memory’s memory 37 is constructed as a hash table over software packages, wherein memory address H denotes the hash value of a software package and that memory location H stores the

number of remaining authorizations M of that software package's use. Ex. 1004, Hellman at 10:33-43; Ex. 1003, Wolfe Declaration at ¶ 142. This architecture is completely incompatible with memory that stores BIOS instructions and memory that is used for BIOS functions like loading, finding, detecting and/or updating the OS.

190. For example, BIOS may store critical information such as the number of cylinders and heads on a computer's attached hard drives or the order in which BIOS will search devices for a bootable operating system.[6] If BIOS memory such as this is also taken to be addressable by hash values and used for storage of authorization counts based solely on the output of a hashing algorithm (hash value H), then the purchase of a software package with an unfortunate hash value H could cause a base unit to overwrite the BIOS instructions or other critical BIOS information with the number of remaining authorizations of that software package's use, rendering the entire system unusable. Thus, any successful use of Hellman's non-volatile memory for BIOS purposes requires abandoning the use that is actually taught by Hellman. Therefore, no POSITA would have combined Hellman and Chou as claimed.

[6] *See* Phil Croucher, "The BIOS Companion," Tri-Tam Enterprises Inc. 1997, pp. 63, 72. (Ex. 2025.)

191. In ¶ 120B, Dr. Wolfe recognizes that "there would have been some risk introduced by storing non-BIOS information in BIOS memory," and attempts to explain it by writing that the "increased risk is what Chou observed as the benefit of doing so in the first place: preventing tampering with the sensitive information without also impacting the BIOS data and thus disabling the entire device. Chou, 1:63-2:7." This is a confusing argument. The relevant text from Chou is as follows. (Only the second paragraph is cited by Dr. Wolfe here.)

> Many computer manufacturers have implemented password protection in the computer BIOS (Basic Input/Output System) which is integral to the operation of a personal computer. The password protection in the BIOS halts the system boot up unless the user enters a password which is also stored in the foregoing CMOS RAM. As noted, if the power is removed from the CMOS RAM, the password is cleared and the system will boot up without requiring the user to enter the required password.
>
> Recent changes in the computer BIOS memory storage devices permit writing data to the BIOS memory, offering the opportunity to provide password protection within the same memory which stores the BIOS routines. Thus, any attempt to delete the protection will result in the BIOS routine being disabled, disabling the boot up process. EEPROM flash devices may be programmed with BIOS routines which permit the user to enter data without requiring the computer to be returned to the manufacture. The present invention makes use of these new BIOS

> memory devices for effecting security measures which discourage theft.

(Ex. 1005 at 1:54-2:7.)

192. In this passage, Chou is not saying that accidentally destroying critical CMOS RAM is an expected consequence of its security measures or even a risk that is somehow worth it under the circumstances. It is simply stating that its techniques are not defeated by removing CMOS RAM power. A user who simply removes CMOS RAM power from a device equipped with Chou's invention would cause it to enter a locked state, from which a user could enter a password without requiring the computer to be returned to the manufacturer. (*Id.* at 7:36-41.) A person who attempts to analogize the CMOS RAM power removal password attack by erasing Chou's EEPROM flash memory would not succeed in gaining access without knowing the password, but instead, would "result in the BIOS routine being disabled, disabling the boot up process." (*Id.* at 1:66-2:1.)

193. Dr. Wolfe attempts to further justify why Hellman's hash-based addressing of its non-volatile memory is compatible with BIOS use of the same memory. In his Ex. 1003 at ¶ 120D, he writes "With Hellman's approach of using a hash value as a memory address, there already would have been some risk of duplicate uses of memory addresses." However, the consequences of accidental collision within Hellman appear to be limited to an incorrect record regarding how

many permitted uses remain for a piece of software. This is quite different than the risk of rendering the entire device inoperable. In ¶120E, Dr. Wolfe suggests enlarging and/or otherwise modifying the memory address range for the hash function output. Concluding in ¶120F, Dr. Wolfe writes "There are likely countless other ways that a POSA would have found it reasonable to implement Hellman as modified by Chou to avoid duplicate use of a memory location by both BIOS data and Hellman's license data." While various approaches can be taken to move things around, the need for such modifications just reinforces that Hellman's non-volatile memory is not a "memory of the BIOS" as required: it is memory for storing counts of allowed uses, addressed by hash value. In contrast, BIOS data is laid out in conventional manner with addresses being allocated for predefined purposes. This is illustrated, for example, in Ex. 1005 [Chou] at Figure 7.

194. Even assuming against the evidence that a POSITA would make the combinations that Dr. Wolfe describes and thereby find some "memory of the BIOS," one cannot conclude that the POSITA would arrive at a system that results in using OS-level software to set up a verification structure in a memory of the BIOS.

195. Indeed, Ewertz was before the U.S. Patent Office during examination and reexamination of the '941 patent. And the Examiner wrote "Ewertz et al. (US 5,479,639) teach the use of BIOS memory for storing licensing numbers." Ex. 1002, '941 file history at 34. In addition, an Examiner considered Lewis (US 5,734,819)

in the *ex parte* reexamination of the ’941 patent. Ex. 2001 at 164. Thus, the use of Chou to find “BIOS memory” is merely cumulative to Ewertz and other art previously considered by the U.S.P.T.O.

3. Alleged Encrypted Verification Structure

196. Dr. Wolfe departs from Hellman, opining that a person of ordinary skill in the art would *not* in fact store M (“the number of uses of software package which are still available”) in the “non-volatile memory 37” as Hellman explicitly teaches, but would store “authorization A” instead. (Ex. 1003 at ¶¶146-147.) In my opinion, a person of ordinary skill in the art would adhere to Hellman’s teaching.

197. To motivate this departure, Dr. Wolfe relies on consequences of his other departures from Hellman’s teaching. Even though Dr. Wolfe states “Hellman does not specifically disclose how update unit 36 is implemented” (*Id.* at ¶ 137), his claim that “a POSA would have recognized that the update unit 36 would have been implemented by a software routine, potentially along with a hardware module” (*Id.* at ¶¶ 137–137A) is crucial to the malicious user attack he envisions—he assumes that a malicious user can write code to “interrogate the non-volatile memory 37, and thereby retrieve the authorized use value M”. (*Id.* at ¶ 142.) But as I explain above, *e.g.* in ¶¶ 164-175, Hellman teaches ***preventing*** access to its security-sensitive areas with physical techniques such as epoxy glue and separate hardware components such as the update unit 36. As discussed above, Schneck similarly relied upon ***hardware***

to protect unencrypted data. (*See*, *e.g.*, Ex. 1006 at 15:51–16:19, 16:49–59.)

198. Dr. Wolfe alternately presents arguments both in the context of Hellman's public-key variant (Ex. 1004 at 11:34-41) and its symmetric key (non-public-key) variant (Ex. 1004 at 9:29-11:19). This alternation is clearest in his analysis of dependent claims; *see* my Section XII.A.5 addressing these dependent claims below. The variant in use affects Dr. Wolfe's opinions on how the "verification structure" of the '941 Patent claim 1 is created and used in the alleged combinations. However, there are significant problems with both variants of the combination, as I explain in the following subsections.

a. Alleged Public-Key Variant

199. The alleged public-key variant of the Hellman/Chou/Schneck combination, used at least in Dr. Wolfe's analysis of claims 1, 2, 3, 6, 11, and 12, [7] simply does not work, as I explain below. In this variant, the secret private key SK is known to the billing and authorization unit 13, which enables it to create authorizations A that can be decrypted using the single public key common to all

[7] Dr. Wolfe does not clearly state which variant he adopts in his analysis of claims 2, 6, and 11. To the extent he opines that a person of ordinary skill in the art would understand the alleged Hellman/Chou/Schneck combination to involve the public-key version, the analysis in this section applies.

base units 12. Since base units do not know the secret private key SK, they are not able to create their own authorizations A. (Ex. 1004 at 11:34-41.) Recall that Hellman teaches that when a user attempts to use a software package, the base unit locates the number M of permitted uses remaining within the non-volatile memory; if M>0, then the use is permitted and the number of remaining permitted uses is decremented to M-1. (*Id.* at 10:33-49.) Thus, according to Hellman's design, when a software package is used, the base unit needs to record the consumption of one "use" by decrementing a counter in the non-volatile memory.

200. But as Dr. Wolfe describes the alleged Hellman/Chou/Schneck combination, the non-volatile memory instead stores an encrypted authorization A that it receives from the billing and authorization unit 13. As an initial matter, this authorization A reflects only the number of *requested additional* uses N, not M+N (the sum of the previously permitted uses and the newly acquired uses), so storing this authorization A in non-volatile memory creates a record with the wrong count. But even worse, without access to the secret private key SK, the base unit 12 has no ability to create and store a necessarily different authorization A' reflecting the situation where a software package has been used and therefore now has fewer

permitted uses remaining.[8]

201. In other words, the system Dr. Wolfe describes is unable to provide the number-of-uses property stated in the first sentence of Hellman's Abstract ("Software (programs, videogames, music, movies, etc.) can be authorized ***for use a given number of times by a base unit after which the base unit*** (computer, video game base unit, record player, videorecorder or video disk player) cannot use that software until the manufacturer sends an authorization for additional uses to the user's base unit") and the first sentence of its column 1 ("The invention relates to a software distribution system and more particularly to a secure software distribution system in which ***the number of uses of software can be controlled***.") Enforcing the "number of uses" constraint requires decreasing the number of remaining uses when the software is used, and the alleged Hellman/Chou/Schneck combination cannot do

[8] I note that part of Hellman's description at 11:29-32 is potentially confusing: "The use of such a public key cryptosystem has the advantage of allowing all users to have the same base unit key in memory 31 and the same secret key SK, which would be the public key." Here, Hellman is teaching that "the same base unit key in memory 31… would be the public key" and that SK is the secret key, *not* that SK itself would be the public key. The description later in this paragraph describing "One secret value, SK, …" confirms that SK is the secret key.

this. This renders the system wholly inoperable for Hellman's intended purpose.

b. Alleged Non-Public-Key Variant

202. The symmetric (non-public-key) variant of the Hellman/Chou/Schneck combination, used at least in Dr. Wolfe's analysis of claims 1, 6-11, 13, 14, and 16,[9] has significant problems as well.

203. Hellman teaches that a request for additional software uses N proceeds as depicted in Figure 5, described at 8:61-9:15. The received authorization A represents four quantities: N, R, SOFTWARE NAME, and BILLING INFORMATION. (Ex. 1004 at 9:10-15.) Hellman extracts these quantities and uses them to store the number of remaining uses as M+N, where M is the base unit's number of remaining uses prior to the request for more. But in Dr. Wolfe's combination that stores A rather than M, this authorization A only expresses the right to N remaining uses, not M+N remaining uses. Short-changing users in this way renders Hellman unsuited to its stated purpose. Dr. Wolfe apparently recognizes this shortcoming and only attempts to justify the behavior when M=0 (no permitted uses

[9] Dr. Wolfe does not clearly state which variant he adopts in his analysis of claims 2, 6, and 11. To the extent he opines that a person of ordinary skill in the art would understand the alleged Hellman/Chou/Schneck combination to involve the non-public-key version, the analysis in this section applies.

remain without further authorization) and therefore M+N=N. (Ex. 1003 at ¶ 170.) Yet Hellman does not teach a method of acquiring N additional uses only when no permitted uses remain; it teaches being able to acquire additional uses no matter how many are currently permitted.

204. Next, Dr. Wolfe asserts that "A POSA would have recognized that storing M in encrypted form would prevent the sort of tampering described above and warned against by Schneck." (*Id.* at ¶ 146.) But his modification does not actually prevent any such tampering *unless* one further assumes that an attacking user has no access to the secret key SK that is used to create valid authorizations. After all, a malicious user who knows SK can use it to create authorizations A, thereby thwarting the alleged protection offered by the combination of Hellman with Schneck. Specifically, Dr. Wolfe does not explain why a user would be able to access Hellman's non-volatile memory directly having redesigned the system for an "update unit 36 in software" (*Id.* at ¶ 137B) but be unable to similarly access the memory containing the secret key SK. To the extent he is making further assumptions about the architecture of the alleged Hellman/Chou/Schneck combination, they are unstated, and any related impacts are unaddressed in his opinions on the motivation to attempt this combination or the likelihood of its success.

205. Finally, in Hellman's teaching, a base unit need only use its symmetric

secret key SK to decrypt the authorization A during “verification of an authorization A to use a software package an additional number of times”, such as part of a purchase transaction, as shown in Fig. 6. (Ex. 1004 at 9:16-21.) The secret SK is not used to determine or report to a base unit user how many permitted uses remain, nor to update the base unit’s number of remaining uses when a software package is used. But Dr. Wolfe’s Hellman/Chou/Schneck combination requires the use of SK for both of these operations, to access the plaintext corresponding to the authorizations A. Dr. Wolfe leaves any security consequences related to the substantially increased use and potential exposure of the secret SK in this combination unaddressed. The acceptability of such increased exposure also appears to rely on Dr. Wolfe’s (unstated) assumption I considered immediately above—that it is easy for an attacker to access Hellman’s non-volatile memory while difficult for an attacker to access SK. Again, the full consequences of his assumptions appear to be left as an exercise for the reader.

4. No Motivation to Combine with Chou

206. Although he does discuss how a person of ordinary skill in the art might be able to select elements from Chou, Dr. Wolfe appears to be silent on any possible motivation that a POSITA would have to combine the fundamentally transformed software-oriented Hellman with Chou in the first instance. This silence makes sense. The purpose of Chou is to discourage computer theft, whereas the purpose of

Hellman is to detect whether software is authorized for use a given number of times by a base unit. Ex. 1005, Chou at Abstract; Ex. 1004, Hellman at Abstract. The benefits of Chou most clearly go to the computer's authorized user, whereas the benefits of Hellman most clearly go to the software publisher. The fact that both systems rely on cryptographic techniques does not justify that a POSITA would be motivated to combine them.

5. Dependent Claims

c. Claim 2

207. Claim 2 is dependent on claim 1, and Dr. Wolfe's analysis of this claim is substantially dependent on his analysis of prior claims. As I explain in Sections XII.A.1-XII.A.4 above, the alleged Hellman/Chou/Schneck combination has significant problems. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent claim 2, including for the reasons discussed above.

d. Claim 3

208. Claim 3 is dependent on claim 2, and Dr. Wolfe's analysis of this claim is substantially dependent on his analysis of prior claims. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent claim 3, including for the reasons discussed above.

209. In ¶ 178, Dr. Wolfe asserts that "a POSA would have found it obvious

to use a public key for the base unit 12 as the 'identification of the computer' in the request for software use. As discussed above for element 3.b, a POSA would have found it obvious to make this modification because it would allow the authorization and billing unit 13 to use the public key as the encryption key for forming authorization A instead of a locally-maintained secret key. For at least those reasons, a POSA would have found it obvious to use 'part of the identification as an encryption key.'"

210. However, Dr. Wolfe appears to have misunderstood Hellman. Hellman does not teach the use of a public key for encryption of messages transferred from the billing unit 13 to the computer; rather, it teaches that its public key is stored in the computer base units 12: "The use of such a public key cryptosystem has the advantage of allowing all users to have the same base unit key in memory 31 and the same secret key SK, which would be the public key."[10] Ex. 1004, Hellman at 11:20-32. For the resulting system to operate at all, this means that the billing unit 13 must encrypt messages that are to be readable by base units using the *private* key. But no private key is transferred from the computer to the bureau, and Dr. Wolfe does not attempt to argue otherwise. He does not even attempt to argue that a public key is transferred from the computer to the bureau. Ultimately, Dr. Wolfe has not identified

[10] *See* footnote 8 on p. 124 above.

any "identification" that is first transferred and then used as part of an encryption key.

211. In addition, as I explain in Sections XII.A.1-XII.A.4 above, the alleged public-key variant of the Hellman/Chou/Schneck combination has significant problems.

e. Claim 6

212. Claim 6 is dependent on claim 1, and Dr. Wolfe's analysis of this claim is substantially dependent on his analysis of prior claims. As I explain in Sections XII.A.1-XII.A.4 above, the alleged Hellman/Chou/Schneck combination has significant problems. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent claim 6, including for the reasons discussed above.

f. Claim 7

213. Claim 7 is dependent on claim 6, and Dr. Wolfe's analysis of this claim is substantially dependent on his analysis of prior claims. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent claim 7, including for the reasons discussed above.

214. I further note that Dr. Wolfe's analysis of claim 7 is done in the context of Hellman's symmetric (non-public-key) version, where "no two users share the same secret key". (Ex. 1003 at ¶193.) As I explain in Sections XII.A.1-XII.A.4

above, the alleged symmetric key variant of the Hellman/Chou/Schneck combination has significant problems.

g. Claim 8

215. Claim 8 is dependent on claim 6, and Dr. Wolfe's analysis of this claim is substantially dependent on his analysis of prior claims. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent claim 8, including for the reasons discussed above.

216. I further note that Dr. Wolfe's analysis of claim 8 is apparently done in the context of Hellman's symmetric (non-public-key) version, wherein "Hellman discloses forming the authorization A by encrypting… using the key SK (which can be the same as key K)". (Ex. 1003 at ¶ 204.) As I explain in Sections XII.A.1-XII.A.4 above, the alleged symmetric key variant of the Hellman/Chou/Schneck combination has significant problems.

h. Claim 9

217. Claim 9 is dependent on claim 7, and Dr. Wolfe's analysis of this claim is substantially dependent on his analysis of prior claims. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent claim 9, including for the reasons discussed above.

218. I further note that Dr. Wolfe's analysis of claim 9 is done in the context of Hellman's symmetric (non-public-key) version, where there is a "unique key K

for the base unit 12”. (Ex. 1003 at ¶ 212.) As I explain in Sections XII.A.1-XII.A.4 above, the alleged symmetric key variant of the Hellman/Chou/Schneck combination has significant problems.

i. Claim 10

219. Claim 10 is dependent on claim 9, and Dr. Wolfe’s analysis of this claim is substantially dependent on his analysis of prior claims. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent claim 10, including for the reasons discussed above.

220. I further note that Dr. Wolfe’s analysis of claim 10 is done in the context of Hellman’s symmetric (non-public-key) version, according to claim 10 being dependent on claim 9. As I explain in Sections XII.A.1-XII.A.4 above, the alleged symmetric key variant of the Hellman/Chou/Schneck combination has significant problems.

j. Claim 11

221. Claim 11 is dependent on claim 1, and Dr. Wolfe’s analysis of this claim is substantially dependent on his analysis of prior claims. As I explain in Sections XII.A.1-XII.A.4 above, the alleged Hellman/Chou/Schneck combination has significant problems. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent claim 11, including for the reasons discussed above.

k. Claim 12

222. Claim 12 is dependent on claim 1, and Dr. Wolfe's analysis of this claim is substantially dependent on his analysis of prior claims. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent claim 12, including for the reasons discussed above.

223. I first note that Dr. Wolfe's analysis of claim 12 is done in the context of Hellman's public-key variant. (Ex. 1003 at ¶ 193.) As I explain in Sections XII.A.1-XII.A.4 above, the alleged public-key variant of the Hellman/Chou/Schneck combination has significant problems.

224. Dr. Wolfe points to the combination of Hellman with Schneck and Chou as disclosing the additional limitation of claim 12, which reads: "A method according to claim 1, wherein a pseudo-unique key is stored in the nonvolatile memory of the BIOS."

225. Note that the "the nonvolatile memory of the BIOS" refers to claim 1's recitation of "an <u>erasable</u>, non-volatile memory area of a BIOS of a computer." However, a POSITA combining Hellman with Chou would not lead to storage of a pseudo-unique K in erasable, non-volatile memory such as EEPROM. Rather, a POSITA motivated to consider moving K from PROM to EEPROM would recognize the security value of leaving it in PROM. This is because Hellman teaches that its key K is stored in "permanent memory 31, for example, a PROM which is

burned in during manufacture of the base unit." Ex. 1004, Hellman at 9:30-32. Moving the key K from PROM to EEPROM enables an attacker to defeat Hellman's authorization limits, because once K is stored in EEPROM, it is much easier for an attacker to copy software (along with its authorizing K) from one device onto another. Furthermore, an attacker could choose an entirely new private key SK', deposit its matching public key K' into a base unit's EEPROM, and use SK' to generate its own authorization values A that would be accepted by the base unit, thereby defeating the intended limitation on software usage altogether.

226. Without acknowledging these problems, Dr. Wolfe speculates at ¶ 227 that it "may be beneficial to change a key periodically, including a public key, such as to prevent a brute force attack attempting to determine the private key being used by the base unit 12" and offers that as a motivation to move the key K from PROM to EEPROM. This suggestion is incorrect. First, in Dr. Wolfe's scenario, the private key is not used by the base unit 12—the base unit uses a *public* key. Ex. 1004, Hellman at 11:20-41. It makes no sense to conduct a "brute force attack" against a public key that can be simply read out of memory. Perhaps Dr. Wolfe meant to address a possible attack against the billing unit 13's *private* key. But if the billing unit 13's private key can be successfully attacked in a world where the base units 12 store their public keys in EEPROM as Dr. Wolfe suggests rather than PROM as Hellman teaches, then the attacker may use any such stolen private key to forge

authorizations and cause base units to accept them by overwriting their public keys stored in their overwritable EEPROM memories. In other words, the attack resistance that Dr. Wolfe sees in being able to "change a key periodically" is much weaker than suggested. In my opinion, a POSITA combining Hellman with Chou would leave Hellman's K in PROM, and this combination does not suggest the requirements of claim 12.

l. Claim 13

227. Claim 13 is dependent on claim 1, and Dr. Wolfe's analysis of this claim is substantially dependent on his analysis of prior claims. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent claim 13, including for the reasons discussed above.

228. I further note that Dr. Wolfe's analysis of claim 13 is done in the context of Hellman's symmetric (non-public-key) version, where "not two users share the same secret key". (Ex. 1003 at ¶ 231.) As I explain in Sections XII.A.1-XII.A.4 above, the alleged symmetric key variant of the Hellman/Chou/Schneck combination has significant problems.

m. Claim 14

229. Claim 14 is dependent on claim 13, and Dr. Wolfe's analysis of this claim is substantially dependent on his analysis of prior claims. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent

claim 14, including for the reasons discussed above.

230. I further note that Dr. Wolfe's analysis of claim 14 is done in the context of Hellman's symmetric (non-public-key) version, where "the authorization A is encrypted using the secret key SK, which can be the same as the key K." (Ex. 1003 at ¶ 234.) As I explain in Sections XII.A.1-XII.A.4 above, the alleged symmetric key variant of the Hellman/Chou/Schneck combination has significant problems.

n. Claim 16

231. Claim 16 is dependent on claim 13, and Dr. Wolfe's analysis of this claim is substantially dependent on his analysis of prior claims. It is my opinion that Dr. Wolfe has not shown that the asserted combination makes obvious dependent claim 16, including for the reasons discussed above.

232. I further note that Dr. Wolfe's analysis of claim 16 is done in the context of Hellman's symmetric (non-public-key) version, according to claim 16 being dependent on claim 13. As I explain in Sections XII.A.1-XII.A.4 above, the alleged symmetric key variant of the Hellman/Chou/Schneck combination has significant problems.

I declare under penalty of perjury that the foregoing is true and accurate to the best of my ability.

Executed on May 3, 2022

[illegible]
Dr. David Martin
Bismarck, North Dakota

David Martin, Ph.D.
3102 Hackberry St.
Bismarck, ND 58503

224 633 9802
dmartin@dmartin.us

May 3, 2022

CURRICULUM VITAE

A. SUMMARY

I have been working in the computer software field for over 40 years, in the roles of developer, designer, professor, and expert consultant. I hold a Ph.D. in computer science and a B.S. in mathematics and computer science.

B. EDUCATION AND ACADEMIC QUALIFICATIONS

1. Education

1999	Ph.D.	Department of Computer Science, Boston University
1993	B.S.	Iowa State University, *summa cum laude* Major: Mathematics Major: Computer Science Minor: German

2. Faculty Experience

2002-2007	Assistant Professor, Department of Computer Science University of Massachusetts Lowell (On leave of absence during 2005-2006)
2001-2002	Research Assistant Professor, Department of Computer Science Boston University
1998-2001	Assistant Professor, Department of Mathematics and Computer Science University of Denver

3. Courses Taught

Computer Networking	Unix Software Tools	Advanced Unix Programming
Cryptography and Network Security	Introduction to Computer Science I (C++)	Introduction to Object Oriented Programming (C++)
Applied Computer Security	Introduction to Computer Science II (C++)	Foundations of Theoretical Computer Science
Special Topics in Systems: Computer Security	Introduction to Computer Science (C)	Formal Languages and Automata

C. Other Professional Activities

1. Government Service

2018-2021	City Councilor, Ames, Iowa (population 66,191 as of 2016; home of Iowa State University of Science and Technology). The Council is the elected government of the City.

2. Other Work Experience

2005-2008	**Cofounder,** Boston Software Forensics. Developed software analysis and protective tools for software publishers and OEMs and provided software forensics consulting services.
1997	**E-mail survey programmer and administrator**, Boston University School of Management, Summer
1996	**Student researcher**, Bellcore Inc., Summer Conducted research in firewall technology, Java security, and cryptography.
1995	**Contract programmer**, BitFlow Inc., Summer Ported device drivers for a digital camera interface board to Windows NT and Windows 95.
1995	**Database and statistical consultant** for Boston University expert witness, Spring
1987-1993	**System programmer and administrator**, Iowa State University. Wrote administration and support code for ISU's distributed network of hundreds of Unix systems. Provided programming and system administration support to Computer Science faculty.
1989	**Student intern programmer**, Hewlett-Packard Germany, Spring Designed and implemented a hypertext help system based on a precursor of Motif for the X Window System under HP Unix while participating in a foreign exchange program.
1986	**Contract programmer**, Lucasfilm Ltd., Fall Designed and implemented a sound sequencer for a computer game in 6502 assembler; arranged music and sound effects for the soundtrack.
1985-1986	**Lead programmer**, Sensor Scan Inc. Designed and implemented object-oriented system and application software for a custom microcomputer-based security control device in 6303 assembler.
1984-1985	**Programmer**, Waveform Corp. Developed software infrastructure for home computer music products in 6502 assembler.

1979-1983	**Part-time programmer**, Cyberia, Inc. Contributed to development of microcomputer software products in BASIC and 6502 assembler, including peripheral-sharing technologies, farm management software, and games.

3. Representative Testifying Expert Engagements (listed by date of initial engagement by underlined party)

2021	e-Numerate Solutions, Inc., and e-Numerate, LLC v. The United States of America (submitted declarations in 2021-22), C.A. No. 19-859-RTH (C. F. C.)
2018	Corus Realty Holdings, Inc. v. Zillow Group, Inc., et al. (submitted expert report; testified at deposition in 2019; W.D. Washington Case No. 2:18-cv-00847-JLR)
2015	ContentGuard Holdings, Inc. v Amazon.com, Inc., et al. (submitted expert report, testified at deposition and trial in 2015; E.D. Tex. Case 2:13-cv-01112-JRG)
2012	Droplets, Inc. v. Overstock.com, Inc., et al. (submitted expert reports; testified at deposition in 2014; testified at trial in 2015; E.D. Tex. Case 2:11-cv-401)
2010	The Apple iPod iTunes Anti-trust Litigation (engaged by plaintiffs; submitted expert reports, testified at depositions in 2011 and 2013; testified at trial in 2014; N.D. Cal. Case 4:05cv00037)
2009	University of California and Eolas Technologies Inc. v. Adobe Systems Inc. et al. (submitted expert reports, testified at depositions in 2011 and 2012, testified at trial in 2012; E.D. Tex. Case 6:2009cv00446)
2008	Northeastern University and Jarg Corporation v. Google, Inc. (submitted expert report, testified at deposition in 2011; E.D. Tex. 2:2007cv00486)
2007	i4i Limited Partnership v. Microsoft Corporation (submitted expert report, testified at deposition in 2009, testified in trial in 2009; E.D. Tex. Case 6:07cv00113)
2004	Lingo et al. (California Consumer Class) v. Microsoft Corporation (submitted expert report, testified at deposition in 2003)

4. Representative Consulting Expert Engagements (listed by date of initial engagement by underlined party)

2020	Match Group, LLC v. Bumble Trading, Inc., Case No. 6: 18-cv-00080-ADA (W.D. Tex.)
2016	Rovi Guides Inc., et al. v. Comcast Corp., et al., No. 2:16-cv-321 (EDTX); Rovi Guides Inc. v. Comcast Corp., et al., No. 2:16-cv-322 (EDTX); Certain Digital Video Receivers and Hardware and Software Components Thereof,

	Inv. No. 337-TA-1001 (ITC); Comcast Corp., et al. v. Rovi Corp., et al., No. 1:16-cv-03852 (S.D. New York)
2015	Zix Corp. v. Echoworx Corp. v. Microsoft Corp., Case No. 2:15-cv-01272-JRG (E.D. Tex.); Case No. 3:2016cv01886 (N.D. Tex.)
2012	VirnetX Inc. and Science Applications International Corporation v. Microsoft Corporation, Civil Action No. 6:13-cv-351 (E.D. Tex.)
2010	TiVo, Inc. v. EchoStar et al.
2009	Prism Technologies LLC v. Research in Motion, Ltd. and Microsoft Corporation
2008	Peer Communications, LLC v. Skype Technologies SA et al.
2008	Sun Microsystems v. Versata Enterprises

5. Memberships

Association for Computing Machinery
National League of Cities Information Technology and Communications Committee

6. Honors and Awards

2007, 2004	Teaching Excellence Award for U. Mass Lowell Computer Science Department
1996	Outstanding Teaching Fellow, Department of Computer Science, Boston University
1993-1994	University Graduate Fellowship, Boston University
1993	Top Graduating Senior in Mathematics, Iowa State University, Spring
1993	Top Graduating Senior in Computer Science, Iowa State University, Spring
1990	Phi Beta Kappa membership (liberal arts honor society)
1990	Phi Kappa Phi membership (engineering honor society)
1990	Pi Mu Epsilon (mathematics honor society)
1990	Upsilon Pi Epsilon (computer science honor society)
1989-1990	Barry M. Goldwater Scholarship (first recipient at Iowa State University)
1988-1989	Participated in Congress-Bundestag Youth Exchange to Germany

7. Academic and Professional Publications

Refereed Publications

2008	Stanley J. Barr, Samuel J. Cardman, and David M. Martin Jr., *A Boosting Ensemble for the Recognition of Code Sharing in Malware*. Journal of Virology 4:4, 2008.
2005	Jesse M. Heines and David M. Martin Jr., *Development and Deployment of a Web-based Course Evaluation System*. In Proceedings of the 2005 International Conference on Web Information Systems and Technologies.
2003	D. Martin, H. Wu, A. Alsaid, *Hidden Surveillance by Web Sites: Web Bugs in Contemporary Use,* Communications of the ACM 46:12ve, December 2003.
2002	D. Martin and A. Schulman, *Deanonymizing Users of the SafeWeb Anonymizing Service*. In Proceedings of the 11th USENIX Security Symposium, August 2002.
2002	A. Alsaid and D. Martin, *Detecting Web Bugs With Bugnosis: Privacy Advocacy Through Education*. In Privacy Enhancing Technologies, Springer Lecture Notes in Computer Science volume 2482, 2002.
2001	J. Burns, P. Gurung, D. Martin, S. Rajagopalan, P. Rao, D. Rosenbluth, and A.V. Surendran, *Automatic Management of Network Security Policy by Self-securing Networks*, Proceedings of DISCEX II, 2001.
2000	D. Martin, R. Smith, M. Brittain, I. Fetch, and H. Wu, *The Privacy Practices of Web Browser Extensions*, Communications of the ACM, February 2001. Extended version available from Privacy Foundation, December 2000.
1998	A. Bestavros, M. Crovella, J. Liu, and D. Martin, *Distributed Packet Rewriting and its Application to Scalable Server Architectures*, Proceedings of the 1998 International Conference on Network Protocols, October 1998.
1997	D. Martin, S. Rajagopalan, and A. Rubin, *Blocking Java Applets at the Firewall*, Proceedings of the Internet Society Symposium on Network and Distributed System Security, January 1997.
1995	R. Book, J. Lutz, and D. Martin, *The Global Power of Additional Queries to Random Oracles*, Information and Computation 120:1, July 1995.

Book Chapters

2004	D. Martin, "Privacy Analysis for the Casual User with Bugnosis", in *Designing Security Systems That People Can Use*, O'Reilly and Associates, 2005.

Book Editor

2005	Privacy Enhancing Technologies, 4th International Workshop, David Martin and Andrei Serjantov (eds.), Lecture Notes in Computer Science 3424, Springer-Verlag 2005.
2006	Privacy Enhancing Technologies, 5th International Workshop, George Danezis and David Martin (eds.), Lecture Notes in Computer Science 3856, Springer-Verlag 2006.

Software Distributions

2001	A. Alsaid, J. Boak, and D. Martin, The Bugnosis Web Bug Detector (Web site and downloadable software), *www.bugnosis.org*, June 2001.

Magazine and Web site articles

2001	D. Martin, "TiVo's Data Collection and Privacy Practices", *Privacy Foundation*, March 2001.
2001	R. Smith and D. Martin, "E-Mail Wiretapping", *Privacy Foundation*, February 2001.
1998	D. Martin, "Internet Anonymizing Techniques", *;login:* Magazine, May 1998.

Technical Reports

2007	S. Barr, S. Cardman, and D. Martin, Matching Global Data References in Related Executables. Computer Science Department Technical Report No. 2007-003.
1999	D. Martin, *Local Anonymity in the Internet*, Ph.D. thesis, May 1999.