

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

ROKU, INC. and VIZIO, INC.,
Petitioner

v.

ANCORA TECHNOLOGIES INC.,
Patent Owner

Case IPR2021-01406
U.S. Patent No. 6,411,941

**DECLARATION OF ANDREW WOLFE, PH.D., IN SUPPORT OF
PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 6,411,941**

Mail Stop "PATENT BOARD"
Patent Trial and Appeal Board
U.S. Patent & Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

Roku EX1003
U.S. Patent No. 6,411,941

TABLE OF CONTENTS

I.	INTRODUCTION	1
A.	Qualifications	2
1.	Education	2
2.	Work Experience.....	2
3.	Curriculum Vitae.....	6
B.	Materials Reviewed.....	7
C.	Level of Ordinary Skill in the Art	7
D.	Summary of Opinions	9
II.	OVERVIEW OF THE TECHNOLOGY.....	9
A.	Priority Date of the Claims.....	9
B.	Overview of Relevant Technology When the '941 Patent Was Filed	10
1.	Software Licenses	10
2.	Computer BIOS.....	12
C.	The '941 Patent	15
D.	Claim Construction.....	22
III.	OVERVIEW OF THE PRIOR ART	22
A.	Hellman	22
B.	Chou	32
C.	Schneck.....	36
IV.	UNPATENTABILITY OF THE '941 PATENT CLAIMS	42
A.	Standards for Invalidity	43
B.	Claim 1	44
1.	Preamble: “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:”	44
2.	Element 1.a: “selecting a program residing in the volatile memory”.....	57
3.	Element 1.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”	61
4.	Element 1.c: “verifying the program using at least the verification structure from the erasable non-volatile memory of the BIOS, and”	71
5.	Element 1.d: “acting on the program according to the verification.”	73
C.	Claim 2: “A method according to claim 1, further comprising the steps of: establishing a license authentication bureau.”	73
D.	Claim 3	74
1.	Preamble: “A method according to claim 2, wherein setting up a verification structure further comprising the steps of:”	74
2.	Element 3.a: “establishing, between the computer and the bureau, a two-way data-communications linkage;”	75
3.	Element 3.b: “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;”	75
4.	Element 3.c: “forming an encrypted license-record at the bureau by encrypting parts of the request-for-license using part of the identification as an encryption key;”	79
5.	Element 3.d: “transferring, from the bureau to the computer, the encrypted license-record; and”	80
6.	Element 3.e: “storing the encrypted license record in the erasable non-volatile memory area of the BIOS.”	81
E.	Claim 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.”	82
F.	Claim 7	83
1.	Preamble: “A method according to claim 6 wherein using an agent to set up the verification structure includes the steps of:”	83
2.	Element 7.a: “establishing or certifying the existence of a pseudo-unique key in a first non-volatile memory area of the computer; and”	83

3.	Element 7.b: “establishing at least one license-record location in the first nonvolatile memory area or in the erasable, non-volatile memory area of the BIOS.”	85
G.	Claim 8	86
1.	Preamble: “A method according to claim 6 wherein establishing a license-record includes the steps of:”	86
2.	Element 8.a: “forming a license-record by encrypting of the contents used to form a license-record with other predetermined data contents, using the key; and”	86
3.	Element 8.b: “establishing the encrypted license-record in one of the at least one established license-record locations.”	87
H.	Claim 9	88
1.	Preamble: “A method according to claim 7 wherein verifying the program includes the steps of:”	88
2.	Element 9.a: “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”	88
3.	Element 9.b: “comparing the encrypted licenses-softwareprogram’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.”	89
I.	Claim 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.”	91
J.	Claim 11: “A method according to claim 1 wherein the volatile memory is a RAM.”	92
K.	Claim 12: “The method of claim 1, wherein a pseudo-unique key is stored in the non-volatile memory of the BIOS.”	92
L.	Claim 13: “The method of claim 1, wherein a unique key is stored in a first non-volatile memory area of the computer.”	95

M.	Claim 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.”	96
N.	Claim 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.”	97
V.	CONCLUSION.....	99

I. INTRODUCTION

1. I, Andrew Wolfe, have been retained by Petitioner Roku, Inc. and Vizio, Inc. (collectively “Petitioner”) to investigate and opine on certain issues relating to United States Patent No. 6,411,941 (“the ’941 patent”) in their Petition for Inter Partes Review of that 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. The opinions set forth in this report are based on my personal knowledge, my professional judgment, and my analysis of the materials and information referenced in this report and its exhibits.

3. I am being compensated for consulting services including time spent testifying at any hearing that may be held. I am also reimbursed for reasonable and customary expenses associated with my work in this case. I receive no other forms of compensation related to this case. My compensation does not depend on the outcome of this inter partes review or the co-pending district court litigation, and I have no other financial interest in this inter partes review.

4. I understand that the ’941 patent has been assigned to Ancora Technologies, Inc.

5. This declaration is based on the information currently available to me. To the extent that additional information becomes available, I reserve the right to

continue my investigation and study, which may include a review of documents and information that may be produced, as well as testimony from depositions that have not yet been taken.

A. Qualifications

1. Education

6. In 1985, I earned a B.S.E.E. degree in Electrical Engineering and Computer Science from The Johns Hopkins University. In 1987, I received an M.S. degree in Electrical and Computer Engineering from Carnegie Mellon University. In 1992, I received a Ph.D. in Computer Engineering from Carnegie Mellon University. My doctoral dissertation proposed a new approach for the architecture of a computer processor.

2. Work Experience

7. I have more than 35 years of experience as a computer architect, computer system designer, personal computer graphics designer, educator, and executive in the electronics industry.

8. In 1983, I began designing touch sensors, microprocessor-based computer systems, and I/O (input/output) cards for personal computers as a senior design engineer for Touch Technology, Inc. During the course of my design projects with Touch Technology, I designed I/O cards for PC-compatible computer systems, including the IBM PC-AT, to interface with interactive touch-based computer terminals that I designed for use in public information systems. I

continued designing and developing related technology as a consultant to the Carroll Touch division of AMP, Inc., where in 1986 I designed one of the first custom touch-screen integrated circuits. I designed the touch/pen input system for the Linus WriteTop, which many believe to be the first commercial tablet computer.

9. From 1986 through 1987, I designed and built a high-performance computer system as a student at Carnegie Mellon University. From 1986 through early 1988, I also developed the curriculum and supervised the teaching laboratory for processor design courses.

10. In the latter part of 1989, I worked as a senior design engineer for ESL-TRW Advanced Technology Division. While at ESL-TRW, I designed and built a bus interface and memory controller for a workstation-based computer system, and also worked on the design of a multiprocessor system.

11. At the end of 1989, I (along with some partners) reacquired the rights to the technology I had developed at Touch Technology and at AMP and founded The Graphics Technology Company. Over the next seven years, as an officer and a consultant for The Graphics Technology Company, I managed the company's engineering development activities and personally developed dozens of touch screen sensors, controllers, and interactive touch-based computer systems.

12. I have consulted, formally and informally, for a number of fabless semiconductor companies. In particular, I have served on the technical advisory boards for two processor design companies: BOPS, Inc., where I chaired the board; and Siroyan Ltd., where I served in a similar role for three networking chip companies—Intellon, Inc., Comsilica, Inc., and Entridia, Inc.—and one 3D game accelerator company, Ageia, Inc.

13. I have also served as a technology advisor to Motorola and to several venture capital funds in the U.S. and Europe. Currently, I am a director of Turtle Beach Corporation, providing guidance in its development of premium audio peripheral devices for a variety of commercial electronic products.

14. From 1991 through 1997, I served on the Faculty of Princeton University as an Assistant Professor of Electrical Engineering. At Princeton, I taught undergraduate and graduate-level courses in Computer Architecture, Advanced Computer Architecture, Display Technology, and Microprocessor Systems, and conducted sponsored research in the area of computer systems and related topics. I was also a principal investigator for DOD research in video technology and a principal investigator for the New Jersey Center for Multimedia Research. From 1999 through 2002, while a Consulting Professor, I taught a Computer Architecture course to both undergraduate and graduate students at Stanford University. At Princeton, I received several teaching awards, both from

students and from the School of Engineering. I have also taught advanced microprocessor architecture to industry professionals in seminars sponsored by the Institute of Electrical and Electronics Engineers (“IEEE”) and the Association for Computing Machinery (“ACM”). I am currently a lecturer at Santa Clara University teaching courses on Microprocessor Systems, Real-Time Computing, and Mechatronics.

15. From 1997 through 2002, I held a variety of executive positions at a publicly-held fabless semiconductor company originally called S3, Inc. and later called SonicBlue Inc. I held the positions of Chief Technology Officer, Vice President of Systems Integration Products, Senior Vice President of Business Development, and Director of Technology, among others. At the time I joined S3, the company supplied graphics accelerators for more than 50% of the PCs sold in the United States. At S3 I supervised the design of several PC graphics accelerators. During my time at SonicBlue we launched more than 30 new consumer electronics products including devices to support copy-protected video and many of the first commercial products to support copy-protected internet audio content.

16. I have published more than fifty peer-reviewed papers in computer architecture and computer systems and IC design. I also have chaired IEEE and ACM conferences in microarchitecture and integrated circuit design and served as

an associate editor for IEEE and ACM journals. I served on the IEEE Computer Society Awards committee. I am a Senior Member of IEEE and a Member of ACM. I am a named inventor on at least fifty-six U.S. patents and thirty-seven foreign patents, which are listed in my curriculum vitae.

17. In 2002, I was the invited keynote speaker at the ACM/IEEE International Symposium on Microarchitecture and at the International Conference on Multimedia. From 1990 through 2005, I have also been an invited speaker on various aspects of technology and the PC industry at numerous industry events including the Intel Developer's Forum, Microsoft Windows Hardware Engineering Conference, Microprocessor Forum, Embedded Systems Conference, Comdex, and Consumer Electronics Show, as well as at the Harvard Business School and the University of Illinois Law School. I have been interviewed on subjects related to computer graphics and video technology and the electronics industry by publications such as the Wall Street Journal, New York Times, Los Angeles Times, Time, Newsweek, Forbes, and Fortune as well as on CNN, NPR, and the BBC. I have also spoken at dozens of universities including MIT, Stanford, University of Texas, Carnegie Mellon University, UCLA, University of Michigan, Rice University, and Duke University.

3. Curriculum Vitae

18. A copy of my *curriculum vitae* is attached as Appendix A to this declaration.

B. Materials Reviewed

19. My opinions expressed in this declaration are based on documents and materials identified in this declaration, including the '941 patent, the prior art references and background materials discussed in this declaration, and the other references specifically identified in this declaration. I have considered these materials in their entirety, even if only portions are discussed here.

20. I have also relied on my own experience and expertise in digital security, software licensing, and computer architecture.

C. Level of Ordinary Skill in the Art

21. I am not an attorney and offer no legal opinions. I have been informed about certain aspects of the law for purposes of my analyses and opinions.¹

¹ I understand that the patent laws were amended by the America Invents Act (AIA), but that the earlier statutory requirements still apply to pre-AIA patents. I have been informed that the '941 Patent is a pre-AIA patent, so the pre-AIA requirements control. Unless otherwise stated, my understanding of the law about patent invalidity as set forth in this declaration relates to the pre-AIA requirements.

22. I understand that in analyzing questions of invalidity, the perspective of a person having ordinary skill in the art (“POSA”) is often implicated, and the Board may need assistance in determining that level of skill.

23. I understand that the claims and written description of a patent must be understood from the perspective of a POSA. I have been informed that the following factors may affect the level of skill of a POSA: (1) the educational level of the inventor; (2) the type of problems encountered in the art; (3) the prior-art solutions to those problems; (4) the rapidity with which innovations are made; (5) the sophistication of the technology; and (6) the educational level of active workers in the field. A person of ordinary skill in the art is also a person of ordinary creativity in the art.

24. Based on my experience in digital security, software licensing, and computer architecture, as well as my reading of the ’941 Patent, it is my opinion 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.

25. I am a person of at least ordinary skill in the art and was so on the date to which the '941 Patent claims priority (May 21, 1998). As shown by my qualifications and my curriculum vitae attached as Appendix A, I am aware of the knowledge and skill possessed by a person of ordinary skill in the art at the time of the alleged invention claimed by the '941 Patent. In performing my analysis, I have applied the standard set forth above.

D. Summary of Opinions

26. I have reviewed and analyzed the '941 Patent (EX1001) as well as prior art references Hellman (U.S. Patent 4,658,093, EX1004), Chou (U.S. Patent 5,892,906, EX1005), and Schneck (U.S. Patent 5,933,498, EX1006).

27. Based on my review and analysis, it is my opinion that claims 1-2, 11, and 13 of the '941 Patent are invalid as obvious based on Hellman in view of Chou. Based on my review and analysis, it is also my opinion that claims 1-3, 6-14, and 16 of the '941 Patent are invalid as obvious based on Hellman in view of Chou and Schneck.

II. OVERVIEW OF THE TECHNOLOGY

A. Priority Date of the Claims

28. I have been informed that a U.S. patent application may claim the benefit of the filing date of an earlier patent application if the earlier patent

application disclosed each limitation of the invention claimed in the later-filed U.S. patent application. I have also been informed that priority is determined on a claim-by-claim basis so that certain claims of a patent may be entitled to the priority date of an earlier-filed patent application even if other claims of the same patent are not entitled to that priority date.

29. I have also been informed that for patent applications filed before March 16, 2013, a patented claim is invalid if the claimed invention was patented or described in a printed publication in any country more than one year before the effective filing date of the claim, regardless of when the applicant conceived of the claimed invention.

30. I understand that the '941 Patent claims a priority date of May 21, 1998.

B. Overview of Relevant Technology When the '941 Patent Was Filed

1. Software Licenses

31. By the time of the '941 Patent's priority date in 1998, the field of software licensing was well-developed. Since at least the 1980s, practitioners in the field had widely recognized the new risks to software piracy introduced by the transformations to digital media.

32. Many entities recognized that one such risk was "copy protection" or "secondary distribution." Secondary distribution contrasted with, for example,

preventing an unauthorized user from obtaining access to a software program in the first place. Secondary distribution dealt with the more challenging problem of allowing a user to have an authorized access to the software program but preventing the user from then making unauthorized copies and distributing those copies. This problem was more challenging because it required some level of trust in the user but balanced against the possibility that the user may still have malicious motivations.

33. For secondary distribution, as with other forms of piracy prevention, encryption was considered a key tool to providing protection. Encryption was a leading solution for various reasons. Encryption was easy to implement but hard to break, making it an efficient solution. Encryption also allowed user-specific and device-specific solutions, given that different devices could be given different encryption/decryption keys.

34. European Patent Application No. EP 0766165A2 (EX1015, “’165 Application”), which published in 1997 from an application filed in 1996, disclosed a license notification system. The ’164 Application disclosed sending encoded license information to a user terminal, with the license information encoded with a key specific to the user terminal. The user terminal checks the license information when the user operates a software program. If the license information is valid, then the licensee’s name is displayed.

35. U.S. Patent No. 5,724,425 (EX1016, “’425 Patent”), which issued in 1998 from an application filed in 1994, disclosed a “software passport.” The software passport was formed by encrypting a message digest using an application writer’s private key, a license, and the software program binary code. A user’s computer uses the encrypted message digest and the license to determine if the software program is secure to operate. The ’425 Patent disclosed this technique to deal with the risk of users purchasing pirated software when they thought they were purchasing legitimate software.

2. Computer BIOS

36. By the time of the ’941 Patent’s priority date in 1998, the field of computer BIOS was well-developed. BIOS began to be used at least as far back as the 1970s, for example in 8-bit computers that ran the CP-M operating system. The usage of BIOS increased rapidly, and by 1998 BIOS was present in essentially all general-purpose computers, e.g., personal computers and servers. In these situations, BIOS provided the basic software routines that were run when the computer was first powered on. One of the primary responsibilities of BIOS was to load the operating system code and allow it to start executing, often called “booting” the computer.

37. For many years, including through to 1998, it was typical to provide BIOS in a separate memory module, apart from the main memory. These came

about for numerous reasons. As one reason, the BIOS programs needed to be secure and away from other program code. Namely, accidentally overwriting or destroying the BIOS program could permanently disable the computer. So storing it on a separate memory module was considered a good approach. As another reason, early versions of BIOS were expected to remain static through the life of the device. As such, it was common to provide BIOS programs in a true read only memory (ROM). By “true” ROM, I mean a memory chip that could not have its contents changed, whether electronically or otherwise. Using true ROM also provided the benefit of not allowing BIOS to be accidentally modified, which was beneficial as described above. Additionally, it was advantageous to provide the BIOS in a non-volatile memory so that it was present when the computer was powered on.

38. By the 1990s, it became more common to store BIOS programs in alterable memory, i.e., memory that could be rewritten. This became more common at least in part because computer manufacturers came to realize that there was a benefit to being able to modify the BIOS programs “in the field,” as opposed to have those programs completely static for the life of the devices.

39. Among these forms of rewritable memory, electrically-erasable programmable read-only memory (EEPROM) was a popular technology. EEPROM was considered beneficial for a number of reasons. For one reason,

EEPROM could be rewritten using simple memory access routines that could be programmatically controlled. This provided the sort of flexibility that computer manufacturers were seeking. For another reason, EEPROM could be implemented as “flash memory,” which was both reliable (not prone to unexpected loss of data) and cost effective (relatively less expensive than some other rewritable ROM technologies).

40. U.S. Patent No. 6,138,236 (EX1017, “’236 Patent”), which issued in 2000 from an application filed in 1996, disclosed the use of both “boot ROM (read only memory)” and “boot PROM (programmable read only memory).” The ’236 Patent explained that the boot PROM could be implemented as flash PROM, “often referred to as flash memory.”

41. U.S. Patent No. 5,802,592 (EX1018, “’592 Patent”), which issued in 1998 from an application filed in 1996, disclosed a technique for verifying the integrity BIOS programs stored in “alterable read only memory (such as FLASH ROM).”

42. U.S. Patent No. 5,835,594 (EX1019, “’594 Patent”), which issued in 1998 from an application filed 1996, disclosed a system for protecting the content, such as BIOS updates, in “FLASH memory or erasable programmable read-only-memory (EPROM).”

C. The '941 Patent

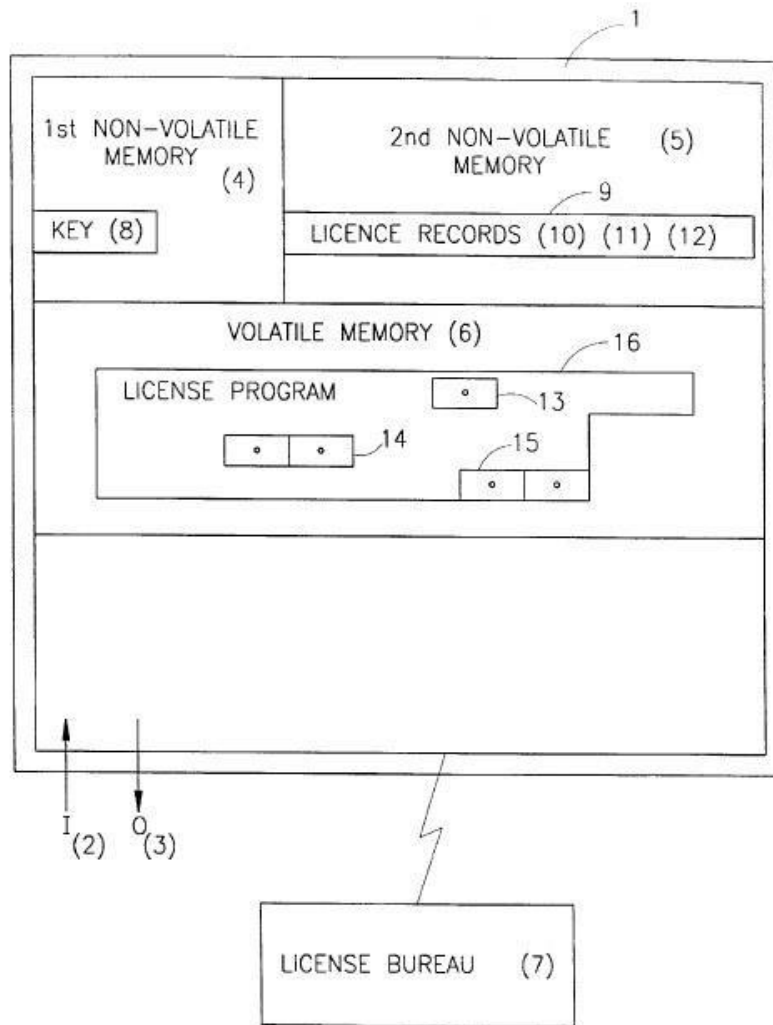
43. The '941 Patent describes a “method of restricting software operation within a license limitation.” '941 Patent, EX1001, Abstract. The '941 Patent explains that there were many known techniques for restricting the operation of an unauthorized software program. EX1001, 1:12-17. The '941 Patent indicates that these techniques were “primarily motivated by the grand proliferation of illegally copied software, which is engulfing the marketplace,” and commented on the large financial impact of this illegal copying. EX1001, 1:12-17.

44. The '941 Patent indicates that one prior art technique involved “writing a license signature onto the computer’s volatile memory (e.g., hard disk).” EX1001, 1:19-26. The '941 Patent explained that this technique was “very vulnerable to attack at the hands of skilled system’s programmers (e.g. ‘hackers’).” EX1001, 1:19-26.

45. The '941 Patent indicates that hardware-based techniques, such as the use of a “dongle” were “expensive, inconvenient, and not particularly suitable for software that may be sold by downloading.” EX1001, 1:27-32.

46. Against that backdrop, the '941 Patent discloses its technique with respect to a computer configuration shown in Figure 1 and a process shown in Figure 2.

47. The computer configuration of Figure 1 (shown below) contains numerous storage devices. The storage devices include the first non-volatile memory area 4, the second non-volatile memory area 5, and the volatile memory area 6. EX1001, Abstract, FIG. 1, 5:9-16. The first non-volatile memory area 4 stores a key 8. EX1001, FIG. 1, 5:19-24. The second non-volatile memory area 5 has a license record area 9 with license records 10, 11, 12. EX1001, FIG. 1, 5:25-33. The volatile memory area 6 include a license program 16, which has license record field 13, 14, and 15. EX1001, FIG. 1, 5:25-33. The computer can communicate with a license bureau 7. EX1001, FIG. 1, 5:17-18.



EX1001, FIG. 1.

48. The '941 Patent provides an example implementation of the invention in a “conventional computer having a conventional BIOS module.” EX1001, 1:43-52. The computer can have a “ROM section” with a key embedded therein at the time of manufacture. EX1001, 1:43-52. “The key constitutes, effectively, a unique identification code for the host computer.” EX1001, 1:43-52. The key cannot be removed or modified. EX1001, 1:43-52.

49. An “application program that is to be licensed to run on the specified computer, is associated with a license record.” EX1001, 1:53-58. “The license record may be held in either encrypted to explicit form.” EX1001, 1:53-58.

50. A license establishment procedure is then performed. EX1001, 1:59-2:9. “[A] verification structure is set in the BIOS so as to indicate that the specified program is licensed to run on the specified computer.” EX1001, 1:592:9. “This is implemented by encrypting the license record ... using said key ... as an encryption key.” EX1001, 1:59-2:9. “The resulting encrypted license record is stored in another (second) non-volatile section of the BIOS, e.g., E²PROM (or the ROM).”² EX1001, 1:59-2:9. The ’941 Patent notes that “unlike the first non-volatile section, the data in the second non-volatile memory may optionally be erased or modified (using E2PROM manipulation commands), so as to enable to add, modify or remove licenses.” EX1001, 1:59-2:9.

51. Once the encrypted license record is stored “in the second non-volatile memory (e.g. E²PROM),” a process for verifying the license can be performed. EX1001, 2:10-2:26. When a “program is loaded into the memory of the computer,” the verification process is performed. EX1001, 2:10-2:26. A license record is retrieved from the program. EX1001, 2:10-2:26. The license record is then

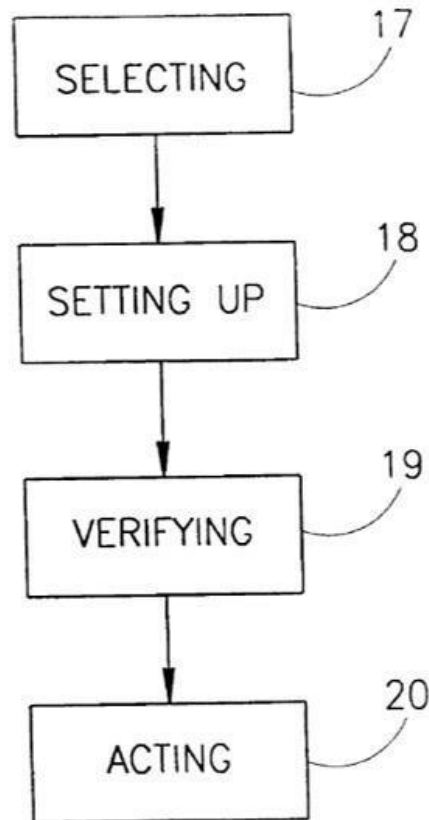
² E²PROM is another spelling of EEPROM.

encrypted using the unique key from ROM. EX1001, 2:10-2:26. That encrypted license record is then compared “to the encrypted records that reside in the E²PROM.” EX1001, 2:10-2:26. If there is a match, then “the program is verified to run on the computer.” EX1001, 2:10-2:26. If there is not a match, then “the program under question is not properly licensed” and appropriate action is taken. EX1001, 2:10-2:26.

52. The '941 Patent describes this technique again with respect to the process of Figure 2, shown below. At step 17, a program is selected. EX1001, 6:7-17. This can include “establishing a licensed-software-program in the volatile memory of the computer. EX1001, 6:7-17. At step 18, the verification structure is set up. EX1001, 6:17-28. This can include “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.” EX1001, 6:17-28. Establishing the license record can include encrypting contents and storing the encrypted license-record in one of the “established license-record locations.” EX1001, 6:17-28. At step 19, the program is verified. EX1001, 6:28-38. This can include encrypting license record contents from the program, and comparing the result with “the encrypted license-record in the first or the second non-volatile memory area.” EX1001, 6:28-38. At step 20, the program is acted on. EX1001, 6:40-52. This can include “restricting the program’s operation with the

predetermined limitations if the comparing yields non-unite or insufficiency.”

EX1001, 6:40-52.



EX1001, FIG. 2.

53. The '941 Patent explains various purported benefits/improvements with the disclosed techniques.

54. The '941 Patent alleges that “[t]hose versed in the art will readily appreciate that any attempt to run a program at an unlicensed site will be immediately detected.” EX1001, 2:28-35. If the program is not licensed on a specific computer, then that will be detected because there will not be an

appropriate license record encrypted with that computer's unique key and stored in the E²PROM. EX1001, 2:28-35.

55. Further, a user cannot subvert this protection by copying a license record in the E²PROM of a first computer to the E²PROM of a second computer. EX1001, 2:37-59. If this were done, then when the verification of the program were performed on the second computer using the key of the second computer, the encryption result would not match the copied license record, which would have been encrypted with the key of the first computer. EX1001, 2:37-59. And the user cannot change the key because it is ROM. EX1001, 2:37-59.

56. The '941 Patent also says that storing the license record in BIOS improves the securing of that information. "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." EX1001, 3:4-17.

"Furthermore, there is a much higher cost to the programmer, if his tampering is unsuccessful, i.e. if data residing in the BIOS (which is necessary for the computer's operability) is inadvertently changed by the hacker. This is too high of a risk for the ordinary software hacker to pay." EX1001, 3:4-17.

D. Claim Construction

57. I understand that claim terms generally are construed in accordance with the ordinary and customary meaning they would have to a POSA at the time of the invention in light of the claim language, the specification, and the prosecution history. I understand that dictionaries and other extrinsic evidence may be considered as well, though such evidence is typically regarded as less significant than the intrinsic record in determining the meaning of the claim language

58. For all terms of the challenged claims of the '941 patent, I have interpreted them as they would have been understood by a POSA at the time of the invention, *i.e.*, May 21, 1998.

III. OVERVIEW OF THE PRIOR ART

A. Hellman

59. Hellman's disclosure, which dates back to its 1983 filing date, describes a system for software distribution. In the system, software (including "programs") can be authorized for a given number of uses on a base unit (including a "computer"). Hellman, EX1004, Abstract. The authorization for additional uses comes from the software's manufacturer. EX1004, Abstract. Hellman discloses a technique whereby the authorization message cannot be reused. EX1004, Abstract. The authorization can be specific to a base unit, "so that an authorization for one base unit cannot be transferred to another base unit." EX1004, Abstract. Hellman

represents that its technique “solves the ‘software piracy problem’.” EX1004, Abstract. As such, Hellman seeks to solve the same problem as the ’941 Patent: using computer-specific authorizations to use a program that prevents unauthorized uses on other computers.

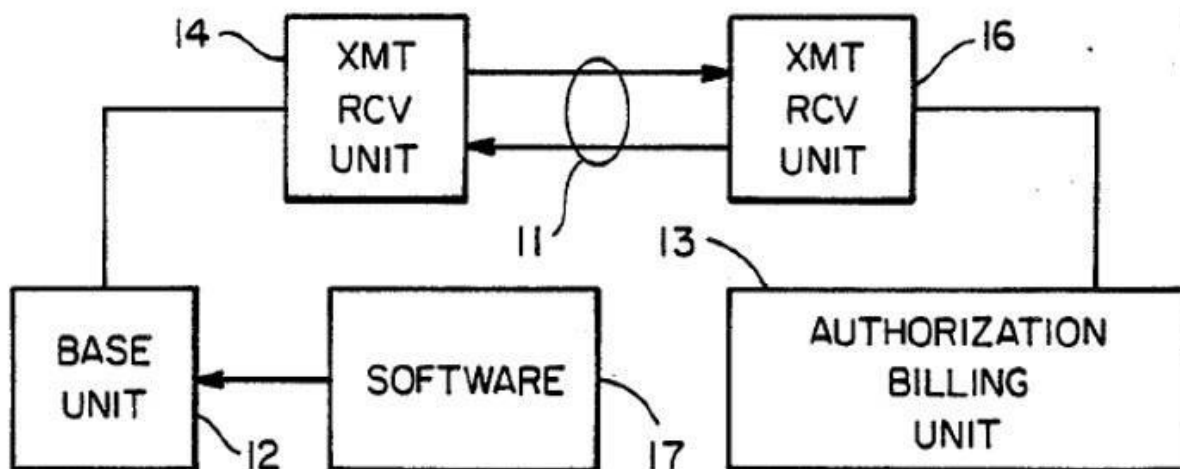
60. Hellman provides more explanation on the “software piracy problem” and the objectives of its disclosure. Hellman explains that “‘software piracy’ is a major problem in the computer and videogame industry.” EX1004, 1:15-16. Hellman discusses a number of existing solutions to software piracy, such as storing a program in a non-standard format and providing a program on a physical disk with a physical defect. EX1004, 1:39-2:6. Hellman says that it would also be beneficial to allow a limit on the number of uses of a program, but that techniques for so limiting program use do not provide for protection against copying once the user has access to the full program. EX1004, 2:7-53.

61. Hellman explains that cryptography is a possible tool for solving these problems, and Hellman discusses various relevant cryptography technologies. EX1004, 2:54-4:21.

62. Hellman sets out three objections to be achieved by its disclosure. First, a software manufacturer should be able to control the number of times a piece of software is used, and the authorization should not be recordable and reusable, and should not be transferable between base units. EX1004, 4:22-27.

Second, software should be able to be sold over telephone or other similar communications channels, without the authorization being reusable on any base unit other than the licensed one. EX1004, 4:28-33. Third, Hellman aims to prevent software piracy, i.e., “the illegal use of software on a base unit which has not paid a license fee.” EX1004, 4:34-36.

63. Hellman depicts the system of its disclosure in Figure 1, shown below. The system includes a base unit 12 and an authorization and billing unit 13 that communicate over an insecure channel 11. EX1004, 5:39-50. The user of base unit 12 obtains “software package 17 by purchasing it at a store, over telephone line, or in some similar manner.” EX1004, 5:51-56.



EX1004, FIG. 1.

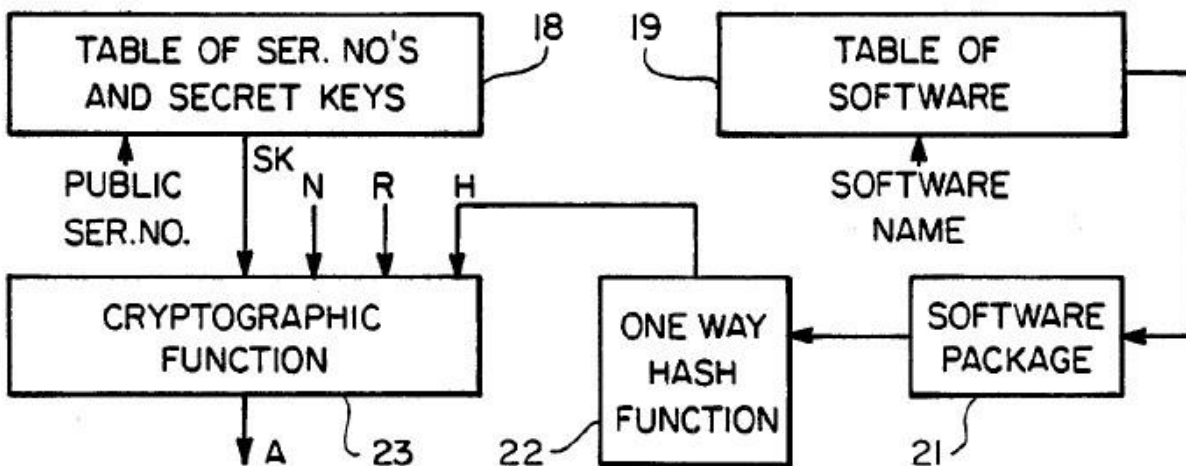
64. The base unit 12 generates a “user originated request for software use” for the software package 17. EX1004, 5:57-6:2. The request contains various

elements of information. EX1004, 5:57-6:2. The request includes a software name, a serial number, a value N, a value R, and billing information. EX1004, 5:57-6:2. The software name is the name of the software package 17. EX1004, 5:57-6:2. The serial number is “a serial number, an identification number, user name or similar identifier unique to base unit 12.” EX1004, 5:57-6:2. The value N is how many additional uses are requested. EX1004, 5:57-6:2. The value R is a “random number, counter value, or other non-repeating number generated by the base unit 12.” EX1004, 5:57-6:2. The billing information is “a credit car[d] number or similar means for billing the user.” EX1004, 5:57-6:2.

65. The base unit 12 transmits the request to the authorization and billing unit 13 over the insecure channel 11. EX1004, 5:57-6:2. The authorization and billing unit 13 receives the request, generates an authorization A “for that particular base unit 12 to use the software package 17 an additional N times” and then transmits the authorization A to the base unit 12. EX1004, 6:3-15. The base unit 12 checks whether the authorization A is correct, and if so updates its memory to allow N additional uses of software package 17. EX1004, 6:3-15.

66. Hellman depicts the operation of the authorization and billing unit 13 in Figure 2, shown below. The authorization and billing unit 13 stores a table of serial numbers and secrets keys in memory 18. EX1004, 6:16-30. The authorization and billing unit 13 uses the serial number received from the base unit

12 to determine the secret key, SK, for the base unit 12. EX1004, 6:16-30. The authorization and billing unit 13 stores a table of software in memory 19 that allows it to determine a software package 21 from the software name provided in the request, and software package 21 is identical to software package 17. EX1004, 6:16-30.



EX1004, FIG. 2.

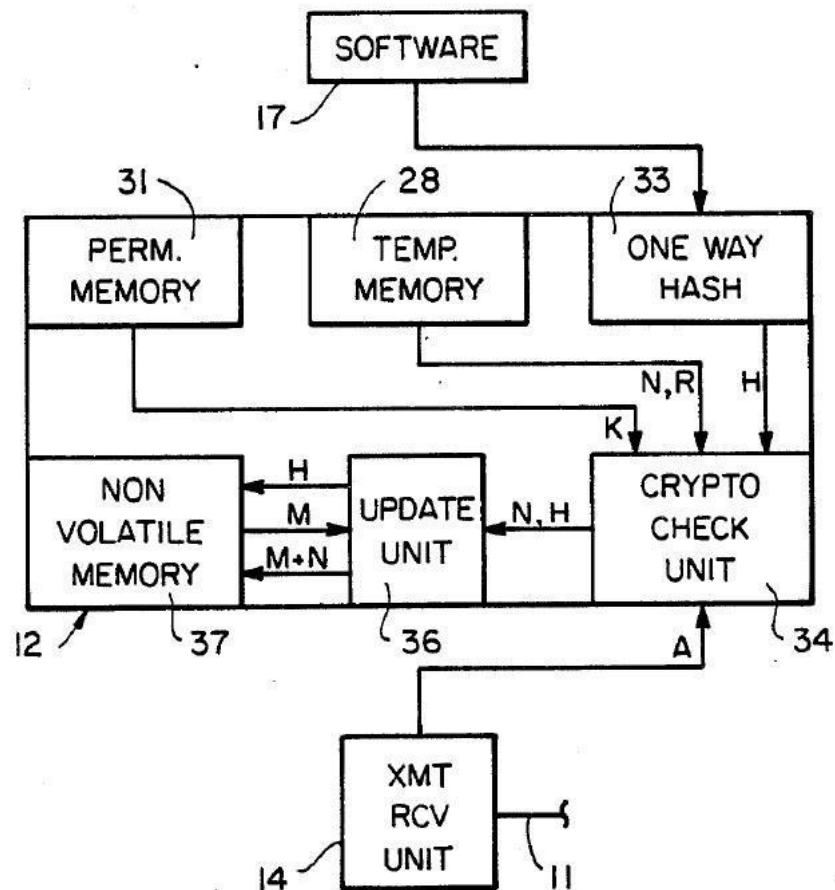
67. A one-way hash function 22 take the software package 21 as input and generates a hash value H. EX1004, 6:31-61. “This output H is used as an ‘abbreviation’ or name for describing the software package 21.” EX1004, 6:31-61. The value H is easily computable from the software package 21 using the one-way has function generator 22, but “given an H value it is difficult, taking perhaps millions of years, to computer any other software package which produces this same H value.” EX1004, 6:31-61. The hash value H is much smaller than the software program 21, with the former containing “perhaps 100 bits,” and the latter

containing “typically 10,000 to 1,000,000 bits.” EX1004, 6:31-61. Storage of the hash value H is preferred to the software name, because, unlike the hash value H, the software name can be easily modified for purposes of circumventing the protections of Hellman. EX1004, 6:31-61.

68. A cryptographic function generator 23 takes as inputs the hash value H, the number of additional uses N, and the random number R, encrypts the inputs using the secret key SK, and thereby generates the authorization A. EX1004, 6:63-7:16. Because the secret key SK is not publicly known, the authorization A can be transmitted over the insecure channel 11 without the risk that it could be decrypted and modified. EX1004, 6:63-7:16. Because the authorization A is encrypted with the secret key SK that is unique to the base unit 12, the authorization A, if intercepted on the insecure channel 11, cannot be reused on another base unit 12 (which would have a different secret key). EX1004, 6:63-7:16. Because the authorization A contains the hash value H, the authorization A, if intercepted on the insecure channel 11, cannot be reused for any other software package (which would have a different hash value). EX1004, 6:63-7:16. Because the authorization A contains the number of uses, N, the authorization A, if intercepted on the insecure channel 11, cannot be reused for a different number of authorized uses. EX1004, 6:63-7:16. Because the authorization A contains the random number, R, the authorization A, if intercepted on the insecure channel 11, cannot be reused for

another request that uses a different random value. EX1004, 6:63-7:16. In this way, even when an authorization to use a software package is transmitted over an insecure channel, the software manufacturer can be sure that the authorization A cannot be reused to allow for other, unauthorized uses of the software package.

69. Hellman depicts the operation of the base unit 12 during verification of authorization A in Figure 6, shown below. The base unit 12 provides the software package 17 as an input to one-way hash generator 33 to generate hash value H, the same hash value generated previously by the authorization and billing unit 13. EX1004, 9:16-28.



EX1004, FIG. 6.

70. The base unit 12 has a key K stored in permanent memory 31.

EX1004, 9:29-40. The permanent memory 31 can be “for example a PROM which was burned in during manufacture of the base unit.” EX1004, 9:7-10. The key K can be the same as the secret key SK used by the authorization and billing unit 13. EX1004, 9:29-40.

71. The base unit has the values N and R stored in temporary memory 28.

EX1004, 9:50-63. The temporary memory 28 can be “for example a RAM.” EX1004, 8:67-68.

72. The base unit 12 operates a cryptographic check unit 34 in order to verify the authorization A. EX1004, 9:50-63. The base unit 12 provides the key K, the value N, the value R, the hash value H, and the received authorization A as inputs to the cryptographic check unit 34. EX1004, 9:50-63. Cryptographic check unit 34 determines that the authorization A is valid if the input of K, N, R, and H results in generation of the same authorization A. EX1004, 9:50-63.

73. If the base unit determines that the authorization A is valid, then the update unit 36 accesses non-volatile memory 37. EX1004, 9:64-10:13. The base unit retrieves the value stored in the memory address represented by hash value H. EX1004, 9:64-10:13. This retrieves the value M, which is the current number of authorized uses remaining. EX1004, 9:64-10:13. The base unit 12 then adds M to

the new number of authorized uses, N., and stores the new total number of authorized uses in non-volatile memory 37 at the address indicated by hash value H. EX1004, 9:64-10:13. The non-volatile memory 37 can be “for example an EEPROM or a CMOS memory with battery backup.” EX1004, 9:64-10:13.

74. Hellman depicts the operation of the base unit 12 upon operation of software package 17 in Figure 6, shown below. When operation of the software package 17 is attempted, the software package 17 is provided as an input to the one-way hash function generator 33 in order to generate the hash value H. EX1004, 10:33-43. The update unit 36 uses the hash value H “as an address to non-volatile memory 37. EX1004, 10:33-43. The non-volatile memory 37 responds by providing the “signal representing M, the number of uses of software package 17 which are still available.” EX1004, 10:33-43.

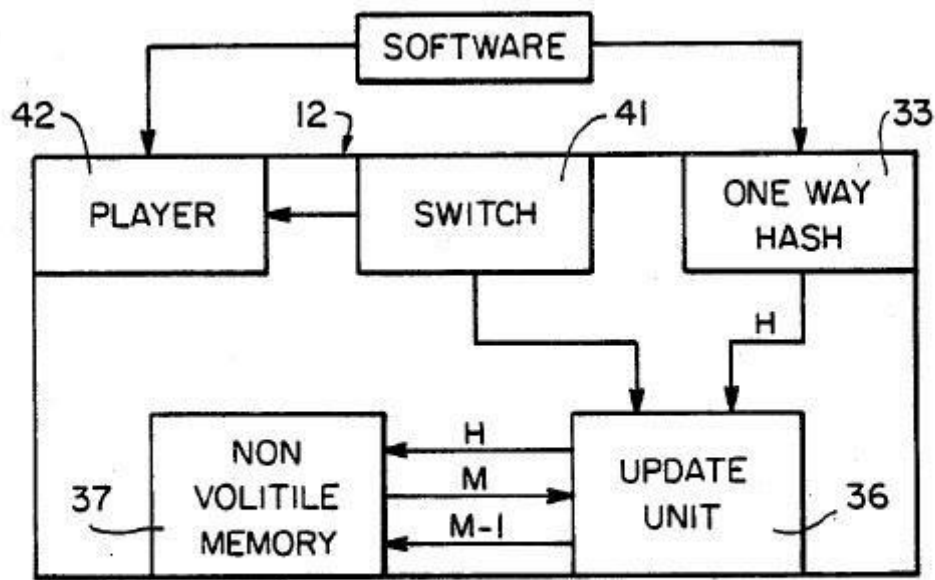


FIG_8

EX1004, FIG. 8.

75. The update unit 36 then determines whether operation of the software package 17 will be permitted. EX1004, 10:44-54. If the value M received from the non-volatile memory 37 is greater than zero, then the update unit 36 uses the switch 41 which “activates software player 42, allowing it to use software package 17.” EX1004, 10:44-54. The update unit 36 then decrements the value M, and “stores this as the new value in address H in non-volatile memory 37.” EX1004, 10:44-54. If the value M retrieved from non-volatile memory 37 is zero, then operation of the software package 17 by the software player 42 is not permitted. EX1004, 10:44-54.

76. Hellman describes a variation to the above disclosure where the software package 17 is authorized for an “unlimited number of uses.” EX1004, 10:55-65. This approach can be implemented “by reserving one value of M to represent infinity.” EX1004, 10:55-65. For example, in an eight-bit value for M, the all 1’s value, 255, can be reserved for “unlimited uses.” EX1004, 10:55-65. “the update unit 36 would be designed to recognize the special pattern of all 1’s and not change it when to software package was used.” EX1004, 10:55-65.

77. The software player 42 “will vary from application to application.” EX1004, 10:66-11:3. “[I]f the software is a computer program, then software player 42 would be a microprocessor or central processing unit (CPU).” EX1004, 10:66-11:3.

B. Chou

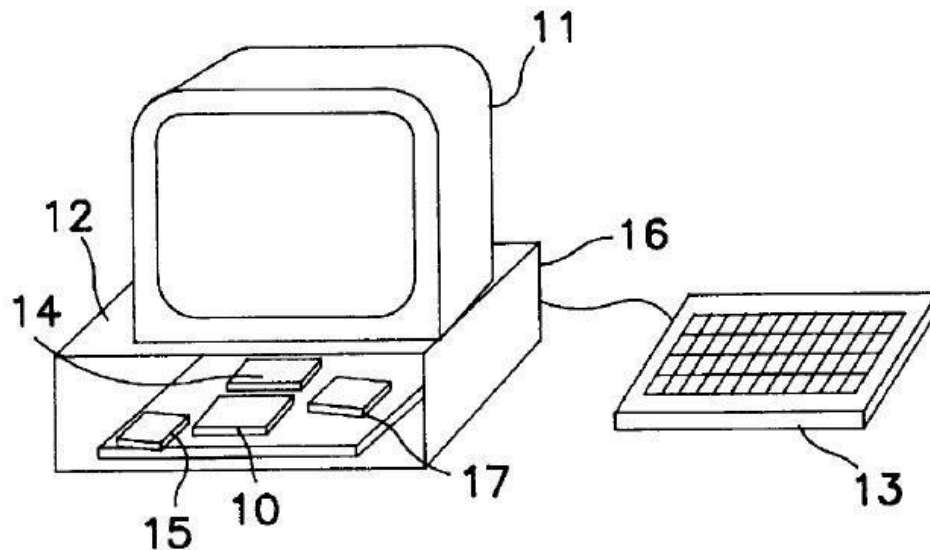
78. Chou’s disclosure, which dates back to a 1996 filing date, describes an approach to discouraging computer theft. Chou, EX1005, Abstract. A security routine is added to the BIOS memory. EX1005, Abstract. The security routine looks for an externally connected memory device or checks for entry of a stored password. EX1005, Abstract. Without passing this security check, the computer will not operate. EX1005, Abstract. The computer can be changed between a mode that requires this security check and a mode that does not require the security check. EX1005, Abstract.

79. Chou explained that an important technological development that motivated the disclosed security techniques was a shift to using writable memory, such as EEPROM, as BIOS memory. EX1005, 1:62-2:7. This created “the opportunity to provide password protection within the same memory which stores the BIOS routines.” EX1005, 1:62-2:7. Chou explained that storing this sensitive information like passwords in the BIOS memory provided a security benefit: “any attempt to delete the protection will result in the BIOS routine being disabled, disabling the boot up process.” EX1005, 1:62-2:7. In other words, Chou made the same observation as the ’941 Patent, but several years earlier: by storing information with BIOS, the risk of rendering the entire computer inoperable discouraged tampering by all but the most advanced hacker. EX1005, 1:62-2:7.

80. Chou disclosed that EEPROM memory was one advantageous way to implement this memory that comingled BIOS routines with other sensitive information. EX1005, 2:2-7. In particular, EEPROM flash devices allowed the user to write data to the memory “without requiring the computer to be returned to the manufacture[r].” EX1005, 2:2-7. Chou disclosed that its invention “makes use of these new BIOS memory devices effecting security measures which discourage theft.” EX1005, 2:2-7.

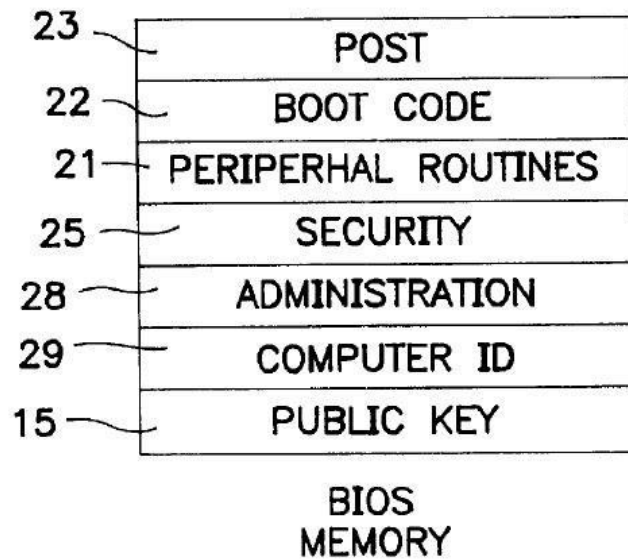
81. Chou disclosed a personal computer 10, as depicted in Figure 1, shown below. The computer 10 included a BIOS EEPROM 15 storing “the BIOS

routines which provide for the basic input/output system.” EX1005, 3:21-28. The BIOS routines also “perform various functions, such as power-on self tests (POST), peripheral routines, boot codes, etc., for initially loading the computer operating system software.” EX1005, 3:44-48. The BIOS EEPROM 15 further stored “a security function stored as a programming routine.” EX1005, 3:21-28.

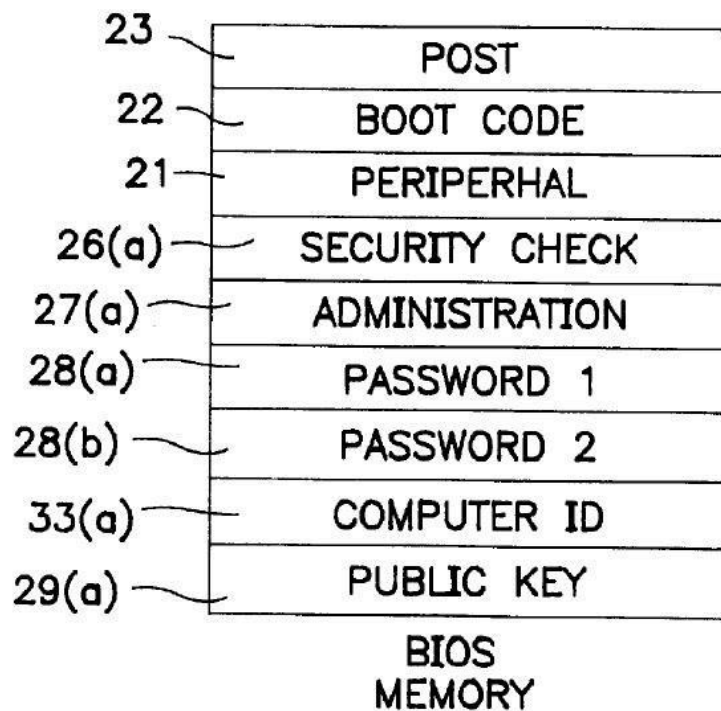


EX1005, FIG. 1.

82. Chou discloses various configurations for the BIOS memory in Figures 3 and 7, shown below. The BIOS memory “may be a flash EEPROM containing various executable BIOS routines as well as routines for implementing a security function.” EX1005, 3:51-67. The BIOS memory can include such BIOS routines as POST (power-on self test) routine 23, boot code 22, and “routine 21 for configuring peripheral devices connected to computer 10.” EX1005, 3:51-67.



EX1005, FIG. 3.



EX1005, FIG. 7.

83. The BIOS memory further includes the security routines 25 that implement the security techniques disclosed by Chou. EX1005, 4:1-5. The security

routine includes information to verify whether the user is an authorized user.

EX1005, FIG. 3, 4:6-19. The BIOS memory stores further information for use in the security routines, such as a computer identifier (alternately referred to as 28, 29, and 33(a), a public key (alternately referred to as 15, 29, and 29(a)), and one or more passwords 28(a) / 28(b). EX1005, FIGs. 3 and 7, 4:6-19, 7:14-35.

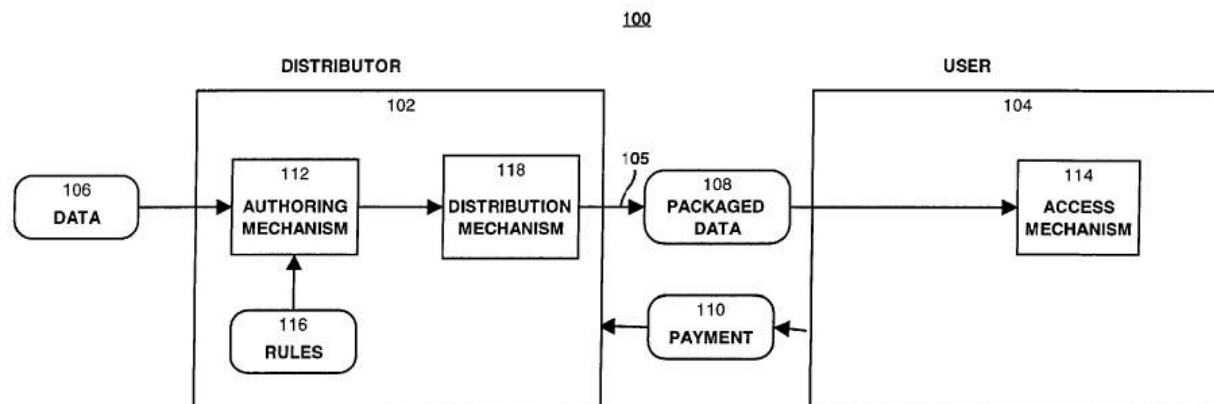
C. Schneck

84. Schneck's disclosure, which dates back to a 1997 filing date and a 1996 priority date, describes an approach to controlling access and distribution of digital data. Schneck, Abstract. Schneck explains that the transition from analog data to digital data greatly increased the ability to create unauthorized copies of data. Schneck, EX1006, 1:40-2:67. Schneck referred to this issue as "secondary distribution" of the digital data. EX1006, 2:46-67.

85. Schneck discloses that cryptography was the "principal technology" for protecting intellectual property. EX1006, 3:30-36. But Schneck observed a problem in the then-existing techniques: "Of those prior art systems which make some use of encryption, none protects the data after it has been decrypted. Thus, secondary distribution and multiple uses are possible." EX1006, 3:57-61. That is, Schneck observes that even if encryption is used to securely transfer digital data or to prevent unauthorized access to digital data, once an authorized access is permitted, a user can make unauthorized copies. EX1006, 3:30-61.

86. Schneck describes that it would be best for control to the digital data to be controlled by the “originator,” similar to the software package manufacturer in Hellman. EX1006, 3:66-67.

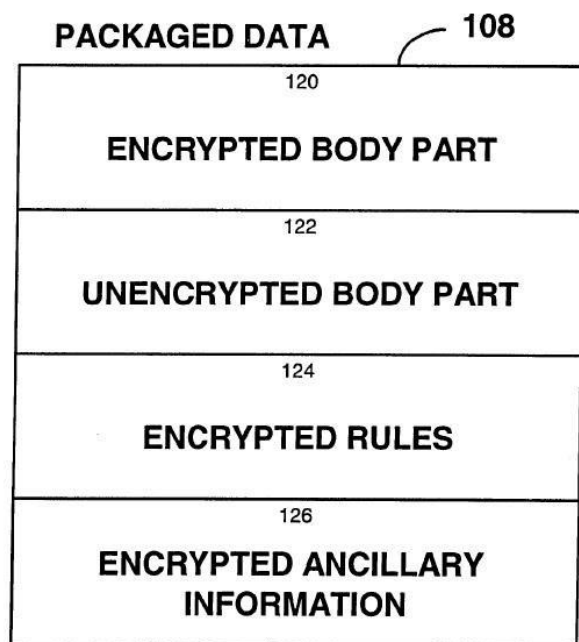
87. Schneck discloses the structure of a system 100 in Figure 1, shown below. The system includes a distributor 102 and a user 104. EX1006, 9:37-59. The distributor takes data 106 as input and uses the authoring mechanism 112 to create packaged data 108. EX1006, 9:37-59. “The packaged data 108 may include access rules 116 in encrypted form encoded therewith, or the access rules 116 may be provided to the user 104 separately.” EX1006, 9:37-59. The distribution mechanism 118 distributes the package data 108 to the user. EX1006, 9:37-59.



EX1006, FIG. 1.

88. Schneck discloses a structure of the packaged data 108 in Figure 2, shown below. The packaged data 108 can included both encrypted body part 120 and unencrypted body part 122. EX1006, 10:35-58. The two body parts are the digital content that the system is protecting (e.g., the data 106). EX1006, 10:35-58.

The packaged data 108 also includes encrypted rules 124, which “are an encrypted version of access rules 116,” and encrypted ancillary information 126. EX1006, 10:35-58. The access rules are encrypted using a “rule-encrypting key” that is “known only to (and protected within) each receiving computer of each user.” EX1006, 12:1-16.



EX1006, FIG. 2.

89. Schneck discloses a structure of the access rules 116 in Figure 3, shown below. The access rules “include various forms of validity checking and identification information.” EX1006, 10:59-11:3. The access rules can include a license number 130. EX1006, 10:59-11:3. The access rules can include encrypted data key 138. EX1006, 10:59-11:3. The access rules can include “the actual rules 140, 142, 144-146 to be Is [sic] applied when access is made to the data by a user.”

EX1006, 10:59-11:3. The actual rules included various permissions and permission lists. EX1006, 10:59-11:3.

116
Version number 127
Authentication (hash) 128
License number of these rules 130
Intellectual property identifier 132
First valid generation of the product 134
Last valid generation of the product 136
Encrypted data key 138
Standard permissions 140
Extended permissions 142
Custom permissions 144
Co-requisite rules (permissions) for source data 145
Token/biometrics 146
System IDs/Public keys 147

EX1006, FIG. 3.

90. Scheck describes additional information for the content of access rules 116 in Table 1, shown below. The access rules 116 can include “System IDs/Public keys” that serve the function of identifying “Other system to which these rules may be redistributed.” EX1006, 11:32-35.

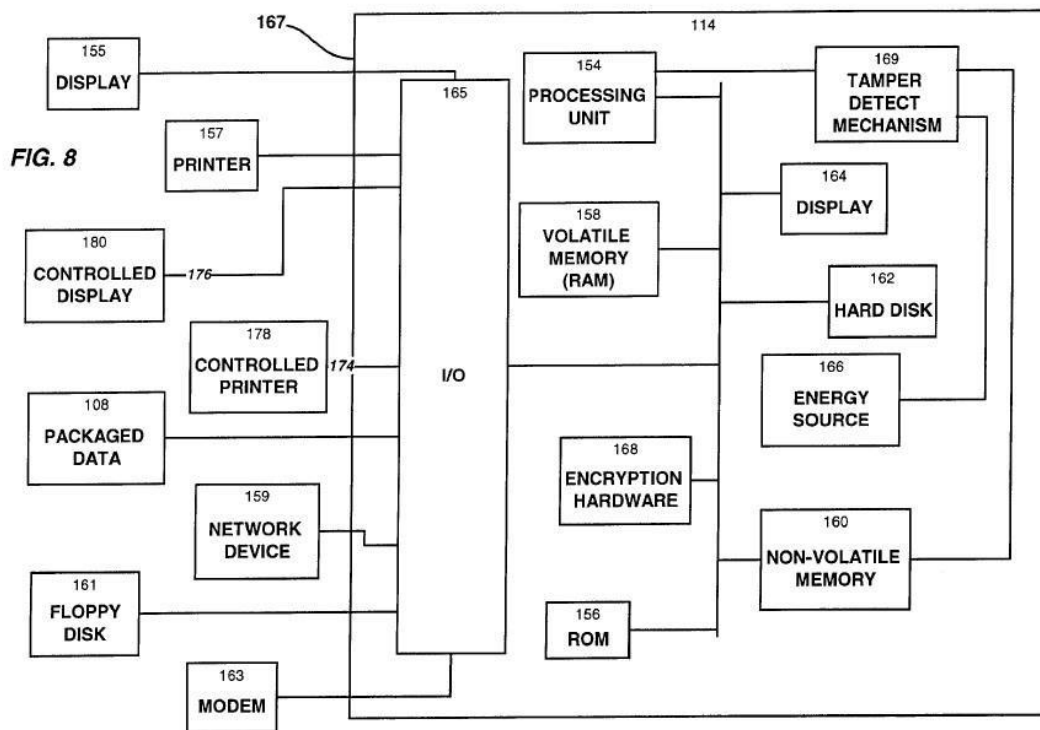
TABLE I

Field	Function
Version number 127	Defines internal configuration template
Authentication (hash) 128	Validates integrity of this data file.
License number of these rules 130.	Used by publisher to identify owner.
Intellectual property identifier 132.	Identifies the intellectual property product.
First valid generation of the product 134.	Defines extent of validity of the license.
Last valid generation of the product 136.	Defines extent of validity of the license.
Encrypted data key 138.	Key to access the data.
Standard permissions 140.	List of basic access permissions for data.
Extended permissions 142.	List of extended access permissions for data.
Custom permissions 144.	Executable code modules.
Co-requisite rules (permissions) for source data 145.	Indicates which source data rules are needed.
Token/biometrics 146	Indicates the physical tokens and/or biometric characteristics (if any) required for identification of each authorized user.
System IDs/Public keys 147	Other systems to which these rules may be redistributed.

91. As noted previously, Schneck allows for the encrypted rules 124 to be provided separate from the packaged data 108. EX1006, 9:51-54. Schneck describes such an embodiment with respect to Figure 5, where the packaged data 150 is transmitted to the user separately from the packaged rules 152. EX1006, 13:37-45. “The packaged data 150 without the rules has the form shown in Fig. 6, which is essentially the same as the structure shown in Fig. 2, but without the encrypted rules 124.” EX1006, 13:37-45.

92. Schneck discloses the structure of the user device with respect to Figure 8, shown below. “The access mechanism 114 allows a user 104 to access

the data in packaged data 108 (or 150) according to the rules provided with (or separately from, as packaged rules 152) the packaged data and prevents the user or anyone else from accessing the data other than as described by the rules.” EX1006, 15:20-29. Schneck notes that the access mechanism can alternatively be provided in a co-processor that operates in cooperation with an existing processing, as shown in Figure 9. EX1006, 16:27-38.



EX1006, FIG. 8.

93. The access mechanism 114 has a number of storage components, such as “volatile memory (RAM) 158,” “electrically-alterable non-volatile memory 160,” a hard disk 162, and read-only memory (ROM) 156. EX1006, 15:30-38.

94. The access 114 mechanism includes a tamper detect mechanism 169.

The purpose of the tamper detect mechanism 169 is to “allow[] the access mechanism 114 to ensure that all internal data (both the system’s data and any user data) are destroyed before any tamperer can obtain them.” EX1006, 16:16-19. When tampering with access mechanism 114 is detected, any cryptographic keys can be destroyed and memory devices can be cleared. EX1006, 15:64-16:15.

95. Schneck disclosed that data stored in non-volatile memory should be stored in encrypted form in order to avoid unauthorized access. EX1006, 16:64-17:5, 17:6-12, 25:64-67. For example: “All communication between the components of the access mechanism 114 and the enclosed hard disk 162 is encrypted. Therefore, if the hard disk is removed the mechanism, any data stored thereon will be inaccessible without the appropriate keys.” EX1006, 16:64-17:5. Further: “In general, within the system, the data are encrypted on any non-volatile storage devices so that they remain unavailable in the case of tampering.” EX1006, 17:6-12.

96. Schneck explains that this encrypting of the data on all non-volatile memory is important to prevent unauthorized secondary distribution: “Since all storage of data on internal non-volatile memory devices (for example, disks, flash memory, and the like) is encrypted, this ensures that a physical attack on the system will not result in compromise of plaintext.” EX1006, 25:64-67.

IV. UNPATENTABILITY OF THE '941 PATENT CLAIMS

A. Standards for Invalidity

97. I am informed and understand that a patent cannot be properly granted for subject matter that would have been obvious to a person of ordinary skill in the art at the time of the alleged invention, and that a patent claim directed to such obvious subject matter is invalid under 35 U.S.C. § 103. It is also my understanding that in assessing the obviousness of claimed subject matter, one should evaluate obviousness in light of the prior art from the perspective of a person having ordinary skill in the art at the time the alleged invention was made (and not from the perspective of either a layman or a genius in that art). It is my further understanding that the question of obviousness is to be determined based on:

- The scope and content of the prior art;
- 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);
- The level of ordinary skill in the art at the time of the alleged invention of the subject matter of the claim; and

- Any relevant objective factors (the “secondary indicia”) indicating nonobviousness, including evidence of any of the following: commercial success of the products or methods covered by the patent claims; a long-felt need for the alleged invention; failed attempts by others to make the alleged invention; copying of the alleged invention by others in the field; unexpected results achieved by the alleged invention; praise of the alleged invention by the alleged infringer or others in the field; the taking of licenses under the patent by others and the nature of those licenses; expressions of surprise by experts and those skilled in the art at the subject matter of the claim; and whether the patentee proceeded contrary to accepted wisdom of the prior art.
- Any relevant objective factors (the “secondary indicia”) indicating obviousness: independent invention of the claimed invention by others before or at about the same time as the named inventor thought of it; and other evidence tending to show obviousness.

B. Claim 1

- 1. Preamble: “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:”**

98. I take no position as to whether this preamble is limiting on the claim, however, it is my opinion that Hellman discloses this feature other than the “erasable, non-volatile memory area of a BIOS.” 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.

99. Hellman discloses a “computer” in the form of the base unit 12. EX1004, FIGs. 1 and 6. Hellman indicates that the base unit 12 can be a computer: “the base unit (computer, videogame base unit, record player, videorecorder or video disk player).” EX1004, Abstract. Hellman also refers to “acquaintances with similar base units (computer).” EX1004, 2:24-29. Hellman discloses that the base unit 12 can have a software player 42 that can be “a microprocessor or central processing unit (CPU).” EX1004, 10:66-11:3. Based at least on these disclosures, a POSA would have recognized that the base unit 12 is a “computer.”

100. Hellman discloses restricting software operation within a license. In particular, Hellman teaches restricting software operation within a license through the disclosure of the number of authorized uses value, M, which is later stored as a license record. Hellman describes M as “the number of authorized uses of the software package ... which still remain unused prior to this new authorization.” EX1004, 9:64-10:13. Hellman discloses that the value M is used to determine if

operation of the software is permitted consistent with previous authorizations.

EX1004, 9:64-10:13, 10:33-49. The value of M is increased based on payment for uses made by the user. EX1004, 9:64-10:13 (incrementing M with new authorized uses N), 5:57-6:15 (receiving authorization for N uses in exchange for billing information). Based at least on these disclosures, a POSA would have recognized that the right to use the software package embodied in the value M reflects “restricting software operation within a license.”

101. Hellman discloses “software” and “software operation” in the form of the software package 17 and its use by the software player 42. Hellman discloses a software package 17 that can be a “computer program.” EX1004, 10:66-11:3. Hellman discloses activating software player 42, “allowing it to use software package 17.” EX1004, 10:44-49. Hellman discloses that the software package 17 can be operated by software play 42, as a computer program being operated by “a microprocessor or central processing unit (CPU).” EX1004, 10:66-11:3. Based at least on these disclosures, a POSA would have recognized that software package 17 is “software” and that use of software package 17 by the software player 42 is “software operation.”

102. Hellman further discloses a “method of restricting software operation within a license” in the form of restricting use of software package 17 by the software player 42 within the limits of authorized uses value, M. Hellman discloses

that the base unit 12 only allows use of the software package 17 by the software player 42 if the value M is greater than zero, and thus more licensed uses are remaining. EX1004, 10:33-65.

103. Hellman discloses that the base unit 12 has a “volatile memory area” in the form of temporary memory 28. Hellman discloses that the base unit 12 has a temporary memory 28. EX1004, 8:66-67, FIG. 6. Hellman discloses that the temporary memory 28 can be provided as RAM. EX1004, 8:66-67. A POSA would have recognized that RAM (an acronym for Random Access Memory) is a type of volatile memory.

104. Hellman discloses that the base unit 12 has an “erasable, non-volatile memory area” in the form of non-volatile memory 37. Hellman discloses that the base unit has a non-volatile memory 37. EX1004, FIG. 6, 9:64-10:13. Hellman discloses that the non-volatile memory 37 can be provided as “an EEPROM.” EX1004, 9:64-10:13. A POSA would have recognized that EEPROM was an acronym for electrically erasable programmable read-only memory. Based at least on these disclosures, a POSA would have recognized that the non-volatile memory 37 was an “erasable, non-volatile memory area.”

105. Hellman does not disclose that the non-volatile memory 37 is a BIOS, or that the base unit 12 included BIOS.

106. Chou disclosed a computer with BIOS and an “erasable, non-volatile memory area of a BIOS.” Chou discloses a “BIOS EEPROM 15.” EX1005, FIG. 1, 3:21-28. Chou discloses that the BIOS memory 15 “may be a flash EEPROM containing the various executable BIOS routines.” EX1005, 3:52-55. Chou discloses the “BIOS Memory” containing various BIOS routines. EX1005, FIGs. 3 and 7. Chou disclosed that there had been a transition from using other types of memory for BIOS to the use of EEPROM. EX1005, 1:63-2:7. Based at least on these disclosures, a POSA would have recognized that the BIOS EEPROM 15 was an “erasable, non-volatile memory area of the BIOS.”

107. A POSA would have found it obvious to include BIOS in the base unit 12 of Hellman. Hellman’s disclosure does not describe BIOS routines, or a memory storing those routines. In my opinion, that lack of disclosure made sense in context, and was not based on any incompatibility between Hellman’s base unit 12 and BIOS.

108. First, while BIOS existed in 1983, BIOS was ubiquitous in all computers at that time. A POSA would have recognized that BIOS was ubiquitous in computers by the time of the May 1998 priority date of the ’941 Patent.

109. Second, while the “BIOS” terminology was used with respect to some general-purpose computers (e.g., microcomputers) in 1983, the terminology was not as consistently used for other types of electronic devices. And Hellman

disclosed its invention generically for multiple types of “base units.” For example, Hellman discloses that the base unit 12 can be any of a “computer,” a “videogame base unit,” a “record player,” a “videorecorder,” and a “videodisk player.”

EX1004, Abstract, 10:66-11:3. Because Hellman’s disclosure was generic to these various types of electronic devices, it is understandable why Hellman did not describe BIOS routines and a BIOS memory for every embodiment. Furthermore, although many non-computer devices at the time stored code in non-volatile memory with the structure and function of a BIOS, they did not generally use that term.

110. Based at least on the foregoing reasons, a POSA would have recognized that Hellman did not disclose BIOS or a BIOS memory in the base unit 12 due to the time of Hellman’s disclosure and the generic nature of the base unit 12. A POSA would have recognized that the lack of a BIOS disclosure in Hellman was not due to any incompatibility of BIOS or BIOS memory with Hellman. In fact, the opposite. A POSA would have understood that at least Hellman’s computer embodiment included a BIOS.

111. By the time of the May 1998 priority date of the ’941 Patent, a POSA would have been aware that BIOS was ubiquitous in general purpose computers (e.g., personal computers). The applicant for the ’941 Patent took an even stronger position during prosecution of the ’941 Patent, saying that “Since all computer

must have a BIOS ...” Office Action dated February 5, 2002, page 7. As a result, when implementing Hellman’s approach as of the May 1998 priority date of ’941 Patent, a POSA would have found it obvious to implement Hellman’s technique in a computer having a BIOS and thus a memory storing the BIOS.

112. When adding BIOS to the base unit 12 of Hellman, a POSA would have found it obvious to use the non-volatile memory 37 of Hellman to store the BIOS routines. There are several independent reasons why a POSA would have been motivated to store the BIOS routines in the non-volatile memory 37 of Hellman.

113. First, a POSA would have been motivated to use non-volatile memory 37 of Hellman to store BIOS because non-volatile memory 37 used a type of memory disclosed by Chou as being advantageous for storing BIOS. Hellman disclosed that non-volatile memory 37 could be provided as EEPROM. EX1004, 9:64-10:13. Hellman disclosed other memory modules, such as permanent memory 31 and temporary memory 28, but Hellman did not disclose any other EEPROM modules. EX1004, 8:61-9:15. Chou disclosed that a transition had occurred to using EEPROM for BIOS, and that the programmability of EEPROM made that an advantageous medium for storing BIOS. EX1005, 1:63-2:7. Based at least on those disclosures, a POSA, when adding BIOS to the base unit 12 of Hellman, would

have been motivated to store the BIOS in the non-volatile memory 37 (i.e., EEPROM) of Hellman.

114. Second, a POSA would have been motivated to use non-volatile memory 37 of Hellman to store BIOS because non-volatile memory 37 would have been one of a limited number of design choices. Hellman disclosed that non-volatile memory 37 could be provided as EEPROM. EX1004, 9:64-10:13. Hellman disclosed other memory modules, such as permanent memory 31 and temporary memory 28, but Hellman did not disclose any other EEPROM modules. EX1004, 8:61-9:15. A POSA would have recognized that in a typical computer as of the May 1998 priority date of the '941 Patent, there would have been a limited number of EEPROM modules available. EEPROM modules were a specialized type of memory not used for general purpose storage. As such, they were generally only added to a computer for a special purpose use. As of May 1998, a typical computer would have had less than five such EEPROM modules, and perhaps only one. As such, the EEPROM module embodied in non-volatile memory 37 of Hellman would have been one of a limited number of choices for where to store the BIOS routines. It would have been easy and obvious to choose non-volatile memory 37 from among this limited set of options. In many computers, the BIOS EEPROM would be the only EEPROM module in the computer.

115. Third, a POSA would have been motivated to use non-volatile memory 37 of Hellman to store BIOS because Chou disclosed that storing sensitive information with the BIOS routines provided extra protection to that sensitive information. Chou disclosed that it was beneficial to store sensitive information like passwords in the BIOS memory because “any attempt to delete the protection will result in the BIOS routine being disabled, disabling the boot up process.” EX1005, 1:63-2:7. A POSA would have understood this disclosure to mean that information should be stored within the BIOS memory because tampering with that information would risk disabling the entire computer. A POSA would have recognized that this would have discouraged many users from attempting to tamper with that information in the first place. Hellman disclosed storing sensitive information, the number of authorized uses M, in non-volatile memory 37. EX1004, 9:63-10:13. Thus, a POSA would have been motivated to store BIOS together with the values M in the non-volatile memory 37, in order to discourage users from tampering with the values M. EX1005, 1:63-2:7.

116. Fourth, a POSA would have been motivated to use non-volatile memory 37 of Hellman to store BIOS because that would have been the most efficient cost and space implementation. As of the May 1998 priority date of the '941 Patent, and still to this day, computer manufacturers generally tried to implement functionality with little cost as possible and in as small of a form factor

as possible. A POSA would have recognized that storing BIOS routines in the already existing non-volatile memory 37 of the base unit 12, as opposed to adding another memory module, would have reduced cost and minimized the physical space occupied by the memory modules.

117. A POSA would have had a strong expectation of success in storing BIOS in the non-volatile memory 37 for numerous reasons.

118. First, Chou already describes storing BIOS in EEPROM, which was one of the formats disclosed for use for non-volatile memory 37. EX1004, 9:64-10:13; EX1005, 1:63-2:7, 3:21-28, 3:62-67.

119. Second, Chou already disclosed that add-on information, such as passwords, encryption keys, and security modules could be added to the memory space used by the BIOS. EX1005, 3:62-4:5, FIGs. 3 and 7. A POSA would have recognized that license information, such as the value M or any other license information, could easily be stored within that BIOS memory space instead.

120. Third, a POSA would have recognized that EEPROM modules of the time had sufficient space to store both the authorization values M from Hellman (which is disclosed in some embodiments as being only 8 bits) and the BIOS routines from Chou. Chou implicitly disclosed this in that it disclosed storing add-on information and security modules in a single EEPROM with BIOS routines. EX1005, 3:62-4:5, FIGs. 3 and 7. A POSA would have recognized that there were

numerous sizes of EEPROM modules available in May 1998 which would have had sufficient space to store the BIOS routines and the authorization values from Hellman.

120A. A POSA would not have been discouraged from the above-described motivations to combine and strong expectation of success based on any alleged incompatibility between the teachings of Hellman and Chou.

120B. As an initial matter, a POSA would have been encouraged by Chou itself to store sensitive information, like Hellman's licensing information, in the BIOS memory. EX1005, 1:63-2:7. While there would have been some risk introduced by storing non-BIOS information in BIOS memory, that 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. EX1005, 1:63-2:7.

120C. Hellman's use of a hash value as a memory address would not have negated this motivation provided by Chou. 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. Namely, with Hellman's approach, one value (i.e., a numerical representation of the software package content itself) was mapped into a second data space (i.e., the maximum range of memory addresses allocated for the purposes of storing the license information in non-volatile

memory 37) using one-way hash function generator 33. A POSA would have recognized that the former value (the software package contents) would have been a larger data space than the latter value (the range of memory addresses). Hellman, 6:48-53. Hence, while perhaps unlikely, it would have been possible for two different software packages to map to hash to the same hash value H and thus the same memory address. This is a fundamental property of hash tables, but in practice is made statistically unlikely enough to be tolerable or insignificant.

120D. Based at least on the foregoing observation, a POSA would have recognized that an implementation of Hellman's approach would have had to account for potential duplicate uses of memory locations. Namely, if two software packages hashed to the same hash value H, Hellman's approach would have need to account for the fact that both would by default have stored license information in the same memory location, one overwriting the other. With that understanding in mind, a POSA would have likewise recognized that an implementation of Hellman's approach where non-volatile memory 37 was the BIOS memory (in light of Chou's disclosure, as discussed above) would also have had to account for potential duplicate uses of memory locations by both BIOS data and a hash value H. A POSA would not have been discouraged by these observations from making the modification of Hellman based on Chou. Rather a POSA would have understood these to be the sort of ordinary and minute implementation details

required by any computer implementation, including, as just discussed, an implementation of Hellman's unmodified disclosure.

120E. As for how a POSA would have avoided duplicate use of a memory location by both BIOS data and a hash value H, there are likely endless such possibilities. As one example, a POSA would have recognized that the one-way hash function generator 33 of Hellman could have been modified to map to a specific range of memory values. Hellman already disclosed this ability, because the one-way hash function generator 33 would already have to be mapping to the range of acceptable memory addresses in non-volatile memory 37. With the combination of Hellman and Chou, a POSA would have found it trivial to modify one-way hash function generator 33 to a different memory address range where the BIOS data was not being stored. For instance, in Chou shows in Figure 3 that a memory space 25 labelled "Security" is allocated in the BIOS Memory 15 in the BIOS space for storing the new security function software disclosed by Chou. A POSA would have recognized that this memory space--potentially enlarged if needed--could be reused for mapping the has values H from the one-way hash function generator 33. In that way, the license information from Hellman would be stored amidst the BIOS data in non-volatile memory 37, but without risk of the license data overwriting the BIOS data.

120F. 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. This is precisely the sort of routine and ordinary design and implementation process that a POSA would be accustomed to performing.

2. Element 1.a: "selecting a program residing in the volatile memory"

121. It is my opinion that Hellman discloses this feature and alternatively renders it obvious based on knowledge of skill in the art. It is my opinion that Schneck further demonstrates that a POSA would have found this feature obvious.

122. Hellman discloses a program in the form of software package 17. EX1004, 5:51-56, 10:50-54, 10:66-11:3. Hellman discloses a volatile memory in the form of temporary memory 28. EX1004, 8:61-9:15.

123. Hellman discloses numerous ways in which the software package 17 is selected. Hellman either discloses explicitly, discloses implicitly, or renders obvious that this selection would be of the software package 17 in the temporary memory 28.

124. First, Hellman discloses that the software package 17 is selected in the form of purchasing the software package 17 "at a store, over telephone line, or in some similar manner." EX1004, 5:51-56. A POSA would have recognized that as part of loading the purchased software package 17, the software package 17 would

have been loaded into temporary memory 28 (e.g., RAM), such as part of transferring the software package 17 from the medium on which it was purchased into some other storage device used by the base unit 12 for storage.

125. Second, Hellman discloses that the software package 17 is selected when the base unit 12 is generating a request for software use. EX1004, 5:57-6:2. As part of generating the request for software use, the base unit 12 determines a software name for the software package 17. EX1004, 5:57-63. A POSA would have recognized that one way in which to determine the software name would have been to load the software package 17 into RAM (temporary memory 28), and extract the software name from the contents of the software package 17.

126. Third, Hellman discloses that the software package 17 is selected when the software package 17 is input into the one-way hash function generator 33. EX1004, 9:16-28, 9:50-63, 10:33-49. The software package 17 is provided as an input to the one-way hash function generator 33 in order to generate the hash value H. EX1004, 9:16-28, 9:50-63, 10:33-49. A POSA would have recognized that the software package 17 would have been present in RAM (temporary memory 28) when it was input to the one-way hash function generator 33. A POSA would have recognized that the one-way hash function generator 33 could have been implemented as either a hardware or a software module. At least in the case where one-way hash function generator 33 was a software module, a POSA would

have recognized that a standard way to provide software package 17 as input to one-way hash function generator 33 would have been to first load software package 17 into RAM, and then provide it as input to the one-way hash function generator 33.

127. Fourth, Hellman discloses that the software package 17 is selected when the software package 17 is selected for use by the software player 42. EX1004, 10:33-11:3. Hellman discloses that software player 42 can “use software package 17.” EX1004, 10:44-49. Hellman discloses that when software package 17 is a “computer program,” the software player 42 is “a microprocessor or central processing unit (CPU).” EX1004, 10:66-11:3. A POSA would have recognized that in this context, the typical way for a software program to be used by a microprocessor or CPU would be to first load it into RAM (temporary memory 28), and then to select it for execution by the microprocessor or CPU.

128. Fifth, a POSA would have found it obvious to modify Hellman so that the software package 17 was maintained in RAM (temporary memory 28) between the time that it was selected for extraction of the software name and the time when it was selected for verifying the authorization A. More specifically, as part of generating the request for software use, the base unit 12 determines a software name for the software package 17. EX1004, 5:57-63. A POSA would have recognized that one way in which to determine the software name would have been

to load the software package 17 into RAM (temporary memory 28), and to extract the software name from the contents of the software package 17. Thereafter, the same software package 17 is provided as an input to the one-way hash function generator 33 in order to generate the hash value H. EX1004, 9:16-28, 9:50-63.

129. A POSA would have recognized that the time between sending the request for software use and the validation of the returned authorization could be a short period of time, e.g., a number of seconds or minutes. A POSA would have recognized that it would have been efficient to maintain the software package 17 in RAM (temporary memory 28) between these two events in order to avoid having to re-load the software package 17 into RAM (temporary memory 28) twice. With such a modification, the software package 17 would still have been in the temporary memory 28 when it is selected for providing as input to one-way hash function generator 33.

130. While this feature was disclosed by or alternatively rendered obvious by Hellman, the feature would further have been obvious based on the teachings of Schneck.

131. Schneck discloses that a first step to accessing protected information would be to “request the operating system to read such data into memory.” EX1006, 18:7–10. The protected information, such as software package 17 from Hellman, would then be selected for any further operations while it was residing in

volatile memory (temporary memory 28). EX1006, 18:7–10. Thus, even if Hellman did not explicitly disclose that the software package 17 was first loaded into temporary memory 28 prior to any of the occasions on which it was selected.

132. A POSA would have been motivated to modify the base unit 12 to first load the software package 17 into temporary memory 28 for several reasons. First, Schneck teaches that this was a standard approach for operating a protected software program, and thus a POSA would have been motivated to use a known and reliable method for operating on software package 17. Second, a POSA would have been motivated to first load the software package 17 into temporary memory 28 because a POSA would have recognized temporary memory 28 as one of a limited number of design choices. In fact, other than temporary memory 28, Hellman does not explicitly describe any other memory devices in the base unit 12 where the software package 17 could be stored during the selecting activities described above. Third, a POSA would have been motivated to first load the software package 17 into temporary memory 28 because a POSA would have recognized that it would have been faster to perform the various selecting activities described above if the software package 17 were present in temporary memory 28 (e.g., RAM), as opposed to some other, slower storage medium.

3. **Element 1.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”**

133. It is my opinion that, based on the modification of Hellman to use non-volatile memory 37 as the “erasable, non-volatile memory of the BIOS” as taught by Chou (discussed above), Hellman discloses this feature. Furthermore, a POSA would have found it obvious to modify Hellman based on the teachings of Schneck to store the number of authorized uses in encrypted form, which is relevant to certain dependent claims.

134. Hellman discloses a “license record” in the form of the number of authorized uses value M. Hellman describes M as “the number of authorized uses of the software package ... which still remain unused prior to this new authorization.” EX1004, 9:64-10:13. Hellman discloses that the value M is used to determine if operation of the software is permitted consistent with previous authorizations. EX1004, 9:64-10:13, 10:33-49. The value of M is increased based on payment for uses made by the user. EX1004, 9:64-10:13 (incrementing M with new authorized uses N), 5:57-6:15 (receiving authorization for N uses in exchange for billing information). In at least some embodiments, M represents unlimited authorized use. Based at least on these disclosures, a POSA would have recognized that the right to use the software package embodied in the value M is a “license record.”

135. 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. Hellman discloses that hash value H is “an ‘abbreviation’ or name for describing the software package 21,” which is an “exact replica” of software package 17. EX1004, 6:16-61.

Hellman discloses that hash value H has the characteristic that “it is easily com[pu]ted from its input signal, software package 21, but given an H value it is difficult, taking perhaps millions of years, to compute any other software package w[h]ich produces this same H value.” EX1004, 6:16-61. Hellman discloses that H is used as an “interrogatory signal” to the non-volatile memory 37, and that update unit 36 uses H “as an address to non-volatile memory 37.” EX1004, 9:64-10:13, 10:33-43.

136. Based at least on these disclosures, a POSA would have recognized that update unit 36 sets up a structure of memory addresses defined by hash value H for storing authorized use values M in the non-volatile memory 37. And because the stored authorized use value M is used to verify if operation of software package 17 is permitted, a POSA would have recognized that this memory structure is a verification structure.

137. Hellman discloses an “agent” in the form of update unit 36. As described above, update unit 36 stores the authorized use value M in the non-volatile memory 37. EX1004, 9:64-10:13. Update unit 36 retrieves the authorized use value M from the non-volatile memory 37. EX1004, 10:33-43. Update unit 36

updates the value M in the non-volatile memory 37. EX1004, 10:44-49. While Hellman does not specifically disclose how update unit 36 is implemented, a POSA would have recognized that the update unit 36 would have been implemented by a software routine, potentially along with a hardware module. Based at least on these disclosures, a POSA would have recognized that the update unit 36 is an agent, and that it sets up the verification structure in the non-volatile memory 37.

137A. A POSA would have recognized that the update unit 36 would have been implemented by software, hardware, or some combination of the two. Hellman does not explicitly say whether the update unit 36 should be implemented in software, hardware, or a combination of the two. A POSA would have recognized from this lack of discussion that it was not necessary that one type of implementation be used over another. In other words, a POSA would have understood that it was up to the discretion of the implementer whether to use software, hardware, or a combination of the two.

137B. This understanding would have been confirmed by the fact that the activities performed by the update unit 36 were of a type that could be performed in software, hardware, or both. The update unit 36 retrieves a value stored at a location in EEPROM, performs integer addition and/or subtraction, and transmits a value to be stored at a location in EEPROM. EX1004, 9:64-10:13. These are all

tasks that a POSA would have understood could be implemented in software, hardware, or both. A POSA would have been motivated to implement the update unit 36 in software in particular because that would have allowed the provider of the base unit to change the implementation logic of the update unit 36 over time, without having to physically disassemble, modify, and reassemble the base unit.

138. A POSA would additionally have recognized that the authorization and billing unit 13 may cooperate with the update unit 36 to act as the “agent.” Hellman discloses that the authorization and billing unit 13 stores a table of serial numbers and secrets keys in memory 18. EX1004, 6:16-30. The authorization and billing unit 13 uses the serial number received from the base unit 12 to determine the secret key, SK, for the base unit 12. EX1004, 6:16-30. The authorization and billing unit 13 stores a table of software in memory 19 that allows it to determine a software package 21 from the software name provided in the request, and software package 21 is identical to software package 17. EX1004, 6:16-30. Because authorization and billing unit 13 generates the authorization A that leads to the updating of the authorized use value M in the non-volatile memory 37, a POSA would have recognized that the authorization and billing unit 13 may be considered an agent.

138A. A POSA would have recognized that the authorization and billing unit 13 would have been implemented by software, hardware, or some combination

of the two. Hellman does not explicitly say whether the authorization and billing unit 13 should be implemented in software, hardware, or a combination of the two. A POSA would have recognized from this lack of discussion that it was not necessary that one type of implementation be used over another. In other words, a POSA would have understood that it was up to the discretion of the implementer whether to use software, hardware, or a combination of the two.

138B. This understanding would have been confirmed by the fact that the activities performed by the authorization and billing unit 13 were of a type that could be performed in software, hardware, or both. The authorization and billing unit 13 stores a table of serial numbers and secret keys, stores a table of information about software packages, performs a hash function, and performs a cryptographic function. EX1004, 6:16-7:16. These are all tasks that a POSA would have understood could be implemented in software, hardware, or both. A POSA would have recognized that the maintaining of the two lookup tables in particular would have been the type of functionality typically involving a software implementation. A POSA would have been motivated to implement the authorization and billing unit 13 in software in particular because that would have allowed the provider of the authorization and billing unit 13 to change the implementation logic of the authorization and billing unit 13 over time, without

having to physically disassemble, modify, and reassemble the authorization and billing unit 13.

139. Furthermore, a POSA would have found it obvious to modify Hellman based on the teachings of Schneck to store the number of authorized uses in encrypted form, which is relevant to certain dependent claims.

140. As an initial observation, Hellman and Schneck attempt to solve a similar problem: preventing an authorized user from distributing unauthorized copies of licensed software. EX1004, 1:39–2:53; EX1006, 2:40–67. Hellman calls this “copy protection,” while Schneck calls this “secondary distribution.” EX1004, 1:39–2:53; EX1006, 2:40–67.

141. While both Hellman and Schneck attempt to address this same problem, Hellman only effectively deals with one aspect of it. Hellman’s approach prevents the interception and reuse of the authorization signal A. EX1004, 6:62–7:16. This thereby protects against the risk of a malicious tampering with respect to the insecure channel 11 between the base unit 12 and the authorization and billing unit 13. EX1004, 6:62–7:16. Hellman achieves this result because the authorization A is encrypted with a key, SK, specific to the base unit 12 and a cryptographic hash specific to the software package 17. EX1004, 6:62–7:16. As a result, the authorization A cannot be reused for a different software package 17 or on a different base unit 12. EX1004, 6:62–7:16.

142. However, a POSA would have recognized that Hellman's approach does not protect against tampering once the authorized use value M is stored in the non-volatile memory 37 of the base unit 12. This is the case because M is stored simply as an integer value, i.e., in plaintext, in the non-volatile memory 37. EX1004, 9:64–10:13. And, the value M is stored at an address determined by hash value H. EX1004, 9:64–10:13. A POSA would have recognized that, as a result, for a software package 17 present on the base unit 12, a malicious user could generate the hash value H, interrogate the non-volatile memory 37, and thereby retrieve the authorized use value M. A POSA could then write a new authorized use value M to the memory address for hash value H, thereby granting new, unauthorized uses to the software package 17. A POSA would have recognized that this tampering was possible because M was stored as a plaintext numerical value, and thus could be overwritten without any need for cryptographic verification.

143. A POSA would have recognized that this risk was heightened for the situation where the number of authorized uses was the special value indicating an “unlimited number of uses of a software package.” EX1004, 10:55-65. Hellman disclosed that a special value of M could be used to indicate infinite uses of the software package 17. EX1004, 10:55-65. Hellman suggests an “all 1's pattern” for this special value. EX1004, 10:55-65. But, a POSA would have recognized that, to

the extent that a malicious user does not know in advance what the special value is, the malicious user could interrogate the non-volatile memory 37 as described above and thereby discover this special infinite use value. A POSA would have recognized that this special value of M could then be stored at the memory address for any software package 17, and thereby grant unlimited uses that were not paid for.

144. Given that Hellman had a shortcoming in its technique for preventing unauthorized secondary distribution, a POSA would have found it obvious to modify Hellman to correct that shortcoming. Schneck discloses one technique that a POSA would have found obvious for correcting this shortcoming of Hellman.

145. Like Hellman, Schneck disclosed that the authorization information, authorization A in Hellman and access rules 116 in Schneck, were transmitted to the user device in encrypted form. EX1006, 9:46–59. But Schneck also disclosed that information that arrives at the user device should be stored in encrypted form, which is the cause of Hellman’s shortcoming. EX1006, 16:64-17:5, 17:6-12, 25:64-67. Schneck discloses that, because “all storage of data on internal non-volatile memory devices (for example, disks, flash memory, and the like) is encrypted, this ensures that a physical attack on the system will not result in compromise of plaintext.” EX1006, 25:64-67.

146. 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. A POSA would have recognized that storing M in encrypted form would prevent the sort of tampering described above and warned against by Schneck.

147. 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. A POSA would have considered this as a desirable solution because base unit 12 already receives authorization A in encrypted form, and thus base unit 12 could simply store the authorization A without the need for additional decrypting/encrypting beyond that already disclosed in Hellman.

148. A POSA would have recognized that storing authorization A in non-volatile memory 37 could have been especially beneficial where the special value M for an unlimited number of uses was used. EX1004, 10:55-65. With the unlimited uses value, the value M never needs to be incremented or decremented. EX1004, 10:55-65. As such, a POSA would have recognized that each time the software package 17 is used, the update unit 36 could simply perform the verification of the value M instead of having to modify M, re-encrypt the value, and then update the storage at address H in non-volatile memory 37.

149. A POSA would have recognized that storing the authorization A in the non-volatile memory 37 would have corrected the vulnerability in Hellman described above. First, because authorization A is encrypted with a key, K, unique to the base unit 12, a malicious user could not copy the authorization A into the non-volatile memory 37 of some other base unit 12 and still have it enable use of software package 17 on that other base unit 12. Second, because authorization A is encrypted with hash value H unique to the software package 17, a malicious user could not copy the authorization A into some other memory address H on the same base unit 12 and have it enable use of the other software package 17.

150. A POSA would have recognized that Hellman's system, when modified based on the teachings of Schneck as described above would achieve the objective of both references of preventing unauthorized secondary distribution of software.

4. Element 1.c: “verifying the program using at least the verification structure from the erasable non-volatile memory of the BIOS, and”

151. It is my opinion that Hellman discloses this feature. As discussed above, Hellman disclosed: the “program” in the form of software package 17; the “verification structure” in the form of the structure of non-volatile memory 37 defined by hash values H storing authorized use values M; and the “erasable non-

volatile memory of the BIOS” in the form of the non-volatile memory 37, as modified by the teachings of Chou to store BIOS routines.

152. Hellman discloses verifying the software package 17 using the verification structure stored in non-volatile memory 37. Hellman discloses that, when use of the software package 17 is attempted, the base unit generates the hash value H. EX1004, 10:33-54. The base unit then uses hash value H as a memory address in non-volatile memory 37 in order to retrieve the number of authorized uses M for that software package 17. EX1004, 10:33-54. The base unit 12 then checks whether the authorized use value M is greater than zero in order to verify whether operation of the software package 17 is permitted. EX1004, 10:33-54. Based at least on these disclosures, a POSA would have recognized that Hellman discloses this feature.

153. When Hellman was modified by the teachings of Schneck to store authorization A in the non-volatile memory 37, a POSA would have recognized that verification would have been performed using the authorization A. A POSA would have recognized that one way in which the verification could be performed would be to decrypt the authorization A using the unique key K of the base unit in order to retrieve the value M included therein. The value M could then be used as already described in Hellman. EX1004, 10:33-54. A POSA would have recognized

that, because the authorization A was encrypted, the authorization A would have to be decrypted in order to retrieve the number of authorized uses value M.

5. Element 1.d: “acting on the program according to the verification.”

154. It is my opinion that Hellman discloses this feature. As discussed above, Hellman disclosed: the “program” in the form of software package 17. Hellman disclosed “the verification” in the form of the verification process described for claim element 1.c.

155. Hellman discloses acting on the program according to the verification in the form of allowing use of the software package 17 by the software player 42 only if the number of authorized uses is greater than zero. EX1004, 10:44-65. If the value of M is greater than zero, then the update unit 36 sends a control signal to the switch 41 to allow software player 42 to use software package 17. EX1004, 10:44-49. If the value of M is zero, then the update unit 36 does not activate the switch to allow the software player 42 to use software package 17. EX1004, 10:50-54. “The user is thus prevented from using software for which he does not have current authorized use.” EX1004, 10:50-54.

C. Claim 2: “A method according to claim 1, further comprising the steps of: establishing a license authentication bureau.”

156. It is my opinion that Hellman discloses this feature.

157. Hellman discloses establishing a “license authentication bureau” in the form of establishing authorization and billing unit 13. Hellman discloses that authentication and billing unit generates an authorization A. EX1004, 6:3-8. Hellman refers to the authorization A as an “authenticator”: “The software manufacturer generates an authenticator which is a cryptographic function of the base unit’s key, the software, the number of times use of the software is authorized, and the random number generated by the base unit.” EX1004, 4:46-63. As discussed previously, a POSA would have recognized that the number of authorized uses is a license because it defines the scope of use of software package 17 that is permitted. EX1004, 6:3-15, 9:64-10:13.

158. As such, the authorization and billing unit 13 generates an authenticator for a license of a software package by encrypting license content for the software package using an encryption key unique to the base unit 12. Based at least on these disclosures, a POSA would have recognized that Hellman discloses “establishing a license authentication bureau.”

D. Claim 3

1. Preamble: “A method according to claim 2, wherein setting up a verification structure further comprising the steps of:”

159. It is my opinion that Hellman discloses this feature, as described for claim 2 above.

2. Element 3.a: “establishing, between the computer and the bureau, a two-way data-communications linkage;”

160. It is my opinion that Hellman discloses this feature.

161. As discussed above, Hellman disclosed: the “computer” in the form of base unit 12; and the “bureau” in the form of the authorization and billing unit 13.

162. Hellman disclosed “establishing ... a two-way data communications linkage” between the base unit 12 and the authorization and billing unit 13 in the form of insecure communication channel 11. EX1004, 5:39-50. Hellman discloses communication over insecure communication channel 11 in one direction when the base unit 12 transmits the request for software use to the authorization and billing unit 13. EX1004, 5:57-6:2. Hellman discloses communication insecure communication channel 11 in the other direction when the authorization and billing unit 13 transmits the authorization A to the base unit 12. EX1004, 6:3-15.

3. Element 3.b: “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;”

163. It is my opinion that Hellman discloses this feature. Furthermore, a POSA would have found it obvious to modify Hellman based on the teachings of Schneck to include additional information in the request-for-license beyond what Hellman discloses.

164. Hellman discloses a “request-for-license” in the form of the “user originated request for software use.” EX1004, 5:57-6:2.

165. Hellman discloses “transferring, from the computer to the bureau” the request for software use, in the form of the base unit 12 transmitting the request for software use to the authorization and billing unit 13. EX1004, 5:57-6:2.

166. Hellman discloses that the request for software use includes “an identification of the computer” in the form of the serial number for the base unit 12. EX1004, 5:57-6:2. Hellman discloses that the request for software use includes a “serial number.” EX1004, 5:57-6:2. Hellman discloses that the serial number can be “a serial number, identification number, user name or similar identifier unique to base unit 12.” EX1004, 5:57-6:2.

167. A POSA would have found it obvious to modify Hellman to use a public key for the base unit 12 in the request for software as the “serial number.” Hellman disclosed that the serial number could be any “serial number, identification number, user name or similar identifier unique to base unit 12.” EX1004, 5:57-6:2. Schneck disclosed that “System IDs/Public keys” can be used to identify a device. EX1006, 11:32-34. A POSA would have been motivated to use a public key in Hellman’s system because Hellman already disclosed that a public key cryptographic system was compatible with Hellman’s system. EX1004, 11:20-68.

168. A POSA would have recognized that the public key could be transmitted from the base unit 12 to the authorization and billing unit 13 as part of the request for software use in order to identify the base unit 12, as suggested by Schneck. EX1006, 11:32-34. A POSA would have recognized the additional benefit of including the public key in the request for software use in order to allow the authorization and billing unit 13 to generate the authorization A without needing to maintain secret keys for the base units 12 in memory of the authorization and billing unit 13. EX1004, 6:16-30. A POSA would have recognized that this would have improved the efficiency and security of the authorization and billing unit 13.

169. Hellman discloses that the request for software use includes “the license-record’s contents from the selected program” in the form of the software name and the number of request uses value N. EX1004, 5:57-6:2. Hellman discloses that the request for software use includes the “software name,” which is “the name of the software package to be used.” EX1004, 5:57-6:2. Hellman discloses that the request for software use includes the value N, which is “the number of additional uses of software requested.” EX1004, 5:57-6:2.

170. At least where the request for software use is the first request for software use for the software package 17, the value N is the value that is ultimately stored in the non-volatile memory 37 as the license record. A POSA would have

recognized that when the first request for software use is transmitted, the existing value M for the number of authorized uses is zero. When the base unit 12 receives the authorization A, which contains the transmitted value N, it is the value N that is stored as the value M in the non-volatile memory 37. EX1004, 9:64-10:13. This is so because “M+N” is equal to the value N when the preexisting value of M is zero. EX1004, 9:64-10:13.

171. A POSA would have found it obvious to modify Hellman to use the hash value H in the request for software use instead of the “software name.” Hellman discloses that the software name “allows authorization and billing unit 13 to determine the complete contents of software package 17 from knowledge of the much smaller information software name.” EX1004, 6:16-30. A POSA would have recognized that in some circumstances, a software name may not be sufficiently unique or well-structured to unique identify the software package 17. For example, if multiple software manufacturers share an authorization and billing unit 13, then there would be a risk that two software manufacturers could inadvertently use the same software name for two different software packages.

172. Based on the risk that the software name would not uniquely identify a software package, a POSA would have been motivated to use the hash value H instead. Hellman discloses that hash value H is an “‘abbreviation’ or name for describing the software package.” EX1004, 6:31-61. As such, Hellman already

indicates that hash value H could be used as a shortened identifier for the software package 17. As such, a POSA would have recognized that the hash value H could serve as an identifier of the software package 17 and would be more likely to uniquely identify the software package 17 than would the software name. With such a modification, the billing and authorization unit could maintain a table correlating hash value H to software package 17, instead of software name to software package 17. EX1004, 6:16-30.

4. Element 3.c: “forming an encrypted license-record at the bureau by encrypting parts of the request-for-license using part of the identification as an encryption key;”

173. It is my opinion that Hellman discloses this feature other than use of “part of the identification as an encryption key.” It is my opinion that a POSA would have found it obvious to include this feature in Hellman’s system based Schneck’s teachings.

174. Hellman discloses forming an “encrypted license-record” in the form of authorization A. As discussed previously for element 1.b, authorization A is a license record. Authorization A is encrypted. EX1004, 6:62-7:2.

175. As discussed previously for claim 2, the authorization and billing unit 13 is a “bureau.” Hellman discloses forming the authorization A at the “bureau” in the form of forming authorization A at the authorization and billing unit 13. EX1004, 6:62-7:2.

176. Hellman discloses encrypting “parts of the request-for-license” to form authorization A in the form of using the value N to form the authorization A. EX1004, 6:62-7:2. The value N is included in the request for software use. EX1004, 5:57-6:2. The value N is encrypted as part of the authorization A. EX1004, 6:62-7:2.

177. As discussed above for element 3.b, a POSA would have found it obvious to include the hash value H in the request for software use. Hellman discloses that the hash value H is included in the authorization A. EX1004, 6:62-7:2. In this additional way, a POSA would have found it obvious to encrypt “parts of the request-for-license” to form authorization A.

178. As discussed above for element 3.b, 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.”

5. Element 3.d: “transferring, from the bureau to the computer, the encrypted license-record; and”

179. It is my opinion that Hellman discloses this feature. As discussed above, Hellman disclosed: the “bureau” in the form of authorization and billing unit 13; the “computer” in the form of base unit 12; and the “encrypted license-record” in the form of authorization A.

180. Hellman discloses transferring the encrypted license-record from the bureau to the computer in the form of transmitting the authorization A from the authorization and billing unit 13 to the base unit 12. EX1004, 6:3-15.

6. Element 3.e: “storing the encrypted license record in the erasable non-volatile memory area of the BIOS.”

181. It is my opinion that, based on the modification of Hellman to use non-volatile memory 37 as the “erasable, non-volatile memory of the BIOS” as taught by Chou (discussed for claim 1 preamble), and based on the modification of Hellman to store the authorization A instead of the plaintext value M (discussed for element 1.b), Hellman discloses this feature.

182. Hellman disclosed storing the value M in the non-volatile memory 37. EX1004, 9:64-10:13. As discussed for claim 1 preamble, a POSA would have found it obvious to use non-volatile memory 37 as the erasable, non-volatile memory of the BIOS.

183. Hellman disclosed storing the value M in the non-volatile memory 37. As discussed for element 1.b, a POSA would have found it obvious to store the

authorization A in the non-volatile memory 37 instead of the plaintext value M.
184.

184. Based on those modification to Hellman discussed previously, a POSA would have found it obvious to store “the encrypted license record in the erasable non-volatile memory area of the BIOS.”

E. Claim 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.”

185. It is my opinion that, based on the modification of Hellman to store the authorization A instead of the plaintext value M (discussed for element 1.b), Hellman discloses this feature.

186. As discussed above, Hellman disclosed: the “computer” in the form of base unit 12; the “volatile memory” in the form of temporary memory 28; a “licensed-software-program” in the form of software package 17; and the “license-record” in the form of authorized uses value M.

187. As discussed for element 1.a, Hellman disclosed loading the software package 17 in the temporary memory 28. Based at least on these disclosures, a POSA would have recognized that Hellman discloses “establishing a licensed-software program in the volatile memory of the computer.”

188. As discussed for element 1.b, a POSA would have found it obvious to store the authorization A in the non-volatile memory 37 instead of the plaintext value M. Hellman discloses that the software package 17 includes contents used to form the authorization A. Hellman discloses that the entirety of the software package 17 is used to generate the hash value H. EX1004, 6:31-61, 9:16-28. The hash value H is part of the authorization A. EX1004, 6:62-7:2. Based at least on these disclosures, a POSA would have recognized that Hellman discloses that the “licensed-software-program includes contents used to form the license-record.”

F. Claim 7

- 1. Preamble: “A method according to claim 6 wherein using an agent to set up the verification structure includes the steps of:”**

189. It is my opinion that Hellman discloses this feature, as described for claim 6 above.

- 2. Element 7.a: “establishing or certifying the existence of a pseudo-unique key in a first non-volatile memory area of the computer; and”**

190. It is my opinion that Hellman discloses this feature

191. As discussed above, Hellman disclosed: the “computer” in the form of base unit 12.

192. Hellman discloses “a first non-volatile memory area” in the form of permanent memory 31. Hellman discloses that base unit 12 includes a “permanent memory 31, for example a PROM.” EX1004, 8:61-9:15. A POSA would have

recognized that PROM is an acronym for programmable read-only memory, and that PROM was a non-volatile memory.

193. Hellman discloses a “pseudo-unique key” in the permanent memory 31 in the form of key K stored in permanent memory 31. EX1004, 9:29-40. Hellman discloses that the key K can be the same as the secret key SK. EX1004, 9:29-40. Hellman discloses that “no two users share the same secret key.” EX1004, 9:41-45.

194. Hellman discloses “certifying the existence” of the key K in the permanent memory 31 in the form of retrieving the key K for input to the cryptographic check unit 34 in order to verify the validity of authorization A. EX1004, 9:50-63, 9:64-10:13, FIG. 6.

195. Based at least on these disclosures, a POSA would have recognized that Hellman discloses “certifying the existence of a pseudo-unique key in a first non-volatile memory area of the computer.”

196. As an additional note, the '941 Patent says that in “the context of the present invention, a ‘pseudo-unique’ key may relate to a bit string which uniquely identifies each first non-volatile memory. Alternately the ‘pseudo-unique’ key may relate to a random bit string (or to an assigned bit string) of sufficient length such that: there is an acceptably low probability of a successful unauthorized transfer of licensed software between two computers, where the first volatile memories of

these two computers have the same key.” EX1001, 4:10-18. Because Hellman discloses that the key K is unique amongst base units 12, a POSA would recognize that the key K is a pseudo-unique key as recited in this claim.

3. Element 7.b: “establishing at least one license-record location in the first nonvolatile memory area or in the erasable, non-volatile memory area of the BIOS.”

197. It is my opinion that Hellman discloses this feature.

198. As discussed above, Hellman disclosed: the “non-volatile memory area of the BIOS” in the form of non-volatile memory 37, as modified by Chou to also store BIOS routines.

199. Hellman discloses “establishing at least one license-record location” in the non-volatile memory 37 in the form of establishing a memory address defined by hash value H where a license record (number of authorized uses M, or authorization A) for the correspond software package 17 will be stored. EX1004, 9:64-10:13. Hellman discloses that a license record for the software package 17 will be stored at a single, specific memory address: the memory address that corresponds to hash value H. EX1004, 9:64-10:13. And hash value H is generated based on the content of the software package itself. EX1004, 31-61, 9:16-28. As such, a POSA would have recognized that Hellman discloses establishing a license record location in the non-volatile memory 37.

200. Based at least on these disclosures, a POSA would have recognized that Hellman discloses “establishing at least one license-record location in the first nonvolatile memory area or in the erasable, non-volatile memory area of the BIOS.”

G. Claim 8

1. Preamble: “A method according to claim 6 wherein establishing a license-record includes the steps of:”

201. It is my opinion that Hellman discloses this feature, as described for claim 6 above.

2. Element 8.a: “forming a license-record by encrypting of the contents used to form a license-record with other predetermined data contents, using the key; and”

202. It is my opinion that, based on the modification of Hellman to store the authorization A instead of the plaintext value M (discussed for element 1.b), Hellman discloses this feature.

203. As discussed above, Hellman disclosed: a “license-record” in the form of authorization A, as modified to Schneck to store authorization A in non-volatile memory 37; the “contents used to form a license-record” in the form of the software package 17 and its corresponding hash value H; and a “key” in the form of key SK.

204. Hellman discloses forming the authorization A by encrypting the “contents used to form a license-record ... using the key” in the form of encrypting

the hash value H with other contents using the key SK (which can be the same as key K). EX1004, 6:62-7:2, 9:29-40.

205. Hellman discloses encrypting “other predetermined data contents” in the license record in the form of number of requested uses N and random number R. EX1004, 6:62-7:2, 9:50-63.

206. Based at least on these disclosures, a POSA would have recognized that Hellman as modified by Schneck discloses “forming a license-record by encrypting of the contents used to form a license-record with other predetermined data contents, using the key.”

3. Element 8.b: “establishing the encrypted license-record in one of the at least one established license-record locations.”

207. It is my opinion that, based on the modification of Hellman to store the authorization A instead of the plaintext value M (discussed for element 1.b), Hellman discloses this feature.

208. As discussed above, Hellman disclosed: an “encrypted license-record” in the form of authorization A, as modified to Schneck to store authorization A in non-volatile memory 37; and the “established license-record locations” in the form the memory addresses defined by a hash value H.

209. Hellman discloses establishing the number of authorized uses value M in the memory location defined by hash value H. EX1004, 9:64-10:13. As discussed above for element 1.b, a POSA would have found it obvious to store the

authorization A in the memory location defined by hash value H, instead of the value M. Based at least on these disclosures, a POSA would have recognized that Hellman as modified by Schneck discloses “establishing the encrypted license-record in one of the at least one established license-record locations.”

H. Claim 9

1. Preamble: “A method according to claim 7 wherein verifying the program includes the steps of:”

210. It is my opinion that Hellman discloses this feature, as described for claim 7 above.

2. Element 9.a: “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”

211. It is my opinion that, based on the modification of Hellman to store the authorization A instead of the plaintext value M (discussed for element 1.b), this feature would have been an obvious modification of Hellman.

212. As discussed above, Hellman disclosed: the “license-record” in the form of authorization A, as modified to Schneck to store authorization A in non-volatile memory 37; the “non-volatile memory area of the BIOS” in the form of the non-volatile memory 37, as modified by Chou to store the BIOS routines; and the “pseudo-unique key” in the form of the key K.

213. As discussed above for element 1.c, when Hellman was modified by the teachings of Schneck to store authorization A in the non-volatile memory 37, a POSA would have recognized that verification would have been performed using the authorization A. A POSA would have recognized that **one** way in which the verification could be performed would be to decrypt the authorization A using the unique key K for the base unit 12 in order to retrieve the value M included therein. The value M could then be used as already described in Hellman. EX1004, 10:33-54. A POSA would have recognized that, because the authorization A was encrypted, the authorization A would have to be decrypted in order to retrieve the number of authorized uses value M.

**3. Element 9.b: “comparing the encrypted licenses-
softwareprogram’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.”**

214. It is my opinion that, based on the modification of Hellman to store the authorization A instead of the plaintext value M (discussed for element 1.b), a POSA would have found this feature to be an obvious modification of Hellman.

214. As discussed above, Hellman disclosed: the “license-software-program's license-record contents” in the form of the software package 17 and its corresponding hash value H; the “license-record” in the form of authorization A, as

modified to Schneck to store authorization A in non-volatile memory 37; and the “non-volatile memory area of the BIOS” in the form of the non-volatile memory 37, as modified by Chou to store the BIOS routines.

215. As discussed above for element 9.a, when Hellman was modified by the teachings of Schneck to store authorization A in the non-volatile memory 37, a POSA would have found it obvious to decrypt the authorization A from the non-volatile memory 37 as part of verification of the software package 17.

216. A POSA would have found it obvious to compare the decrypted authorization A with the software package 17 and the corresponding hash value H. As discussed above for element 1.b, one of the reasons that a POSA would have been motivated to modify Hellman to store the authorization A instead of the number of authorized uses value M in the non-volatile memory 37 was the fact that storing plaintext value M would allow a malicious user to change the value of M for any software package 17. Further, as discussed above for element 1.b, a POSA would have recognized that storing authorization A would have prevented this tampering, because the hash value H was part of the authorization A, and as such the authorization A could not be reused at any other memory location in the non-volatile memory 37.

217. A POSA would have found it obvious that in order to achieve this benefit in the modification of Hellman, it would be necessary to compare the hash

value H included as part of the authorization A to the software package 17 that the user was attempting to use. If the base unit 12 did not perform this comparison of the hash value H in the authorization A to the hash value H for the software package 17 in use, then the tampering described above would not be prevented. As such, a POSA would have found it obvious to implement this comparison in the modified system of Hellman, and thereby include “comparing the license-software-program's license-record contents with the decrypted license-record in erasable non-volatile memory area of the BIOS.”

I. Claim 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.”

218. It is my opinion that Hellman discloses this feature. As discussed above, Hellman disclosed: the “program” in the form of software package 17.

219. Hellman discloses restricting the operation of the software package 17 in the form of preventing the user from using the software package 17 when the number of authorized uses equals zero. EX1004, 10:50-54.

220. As discussed above for element 9.b, a POSA would have found it obvious to compare the hash value H from the authorization A stored in the non-volatile memory 37 to the hash value H for the software package 17 that the user is attempting to use. A POSA would have recognized that when those hash values H do not match, the authorization A does not correspond to the software package 17

that the user is attempting to use. In such a case, a POSA would have found it obvious to prevent the user from using the software package 17. EX1004, 10:50-54. A POSA would have been motivated to prevent this operation in order to prevent the user from making unauthorized use of the software package 17.

J. Claim 11: “A method according to claim 1 wherein the volatile memory is a RAM.”

221. It is my opinion that Hellman discloses this feature. As discussed above for claim 1 preamble, Hellman disclosed the “volatile memory” in the form of temporary memory 28. Hellman disclosed that temporary memory could be “for example a RAM.” EX1004, 8:67-68.

K. Claim 12: “The method of claim 1, wherein a pseudo-unique key is stored in the non-volatile memory of the BIOS.”

222. It is my opinion that this feature would have been an obvious modification of Hellman based on the teachings of Schneck. As discussed above, Hellman disclosed: the “non-volatile memory of the BIOS” in the form non-volatile memory 37, as modified by Chou to store BIOS routines.

223. A POSA would have found it obvious to modify Hellman to use a public key for the base unit 12 in the request for software as the “serial number.” Hellman disclosed that the serial number could be any “serial number, identification number, user name or similar identifier unique to base unit 12.” EX1004, 5:57-6:2. Schneck disclosed that “System IDs/Public keys” can be used

to identify a device. EX1006, 11:32-34. A POSA would have been motivated to use a public key in Hellman's system because Hellman already disclosed that a public key cryptographic system was compatible with Hellman's system. EX1004, 11:20-68.

224. A POSA would have recognized that the public key could be transmitted from the base unit 12 to the authorization and billing unit 13 as part of the request for software use in order to identify the base unit 12, as suggested by Schneck. EX1006, 11:32-34. A POSA would have recognized the additional benefit of including the public key in the request for software use in order to allow the authorization and billing unit 13 to generate the authorization A without needing to maintain secret keys for the base units 12 in memory of the authorization and billing unit 13. EX1004, 6:16-30. A POSA would have recognized that this would have improved the efficiency and security of the authorization and billing unit 13.

225. A POSA would have recognized that the public key would be a "pseudo-unique key." Schneck disclosed that "System IDs/Public keys" can be used to identify a device. EX1006, 11:32-34. Based on this disclosure of Schneck and the knowledge of public key cryptography in the art as of May 1998, a POSA would have recognized that the public key would be able to identify a particular base unit 12.

226. When Hellman was modified to use a public key, a POSA would have recognized that the public key could be stored in the non-volatile memory 37. A POSA would have recognized that the public key, like any other key disclosed in Hellman, would need to be stored in non-volatile memory in order to be maintained when power was lost. EX1004, 9:16-49. Hellman only disclosed two such non-volatile memories: permanent memory 31 and non-volatile memory 37. EX1004, 8:61-9:15, FIG. 6. A POSA would have found it obvious to store the public key in the non-volatile memory 37 as one of a limited number of design choices, i.e., one of two options.

227. Additionally, a POSA could have found it obvious to store the public key in the non-volatile memory 37 in order to allow that value to change. Hellman disclosed that the key K was stored in permanent memory 31 because it would not change. EX1004, 9:16-49. But a POSA would have recognized 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. A POSA would recognize that, in order for the public key to be able to change, it would be beneficial to public key in the non-volatile memory 37.

228. A POSA would have further found it obvious to store the public key in the non-volatile memory 37 because Chou discloses that a public key can be stored in the BIOS memory. EX1005, 4:6-19, FIGs. 3 and 7. And, as discussed

above for claim 1 preamble, a POSA would have found it obvious to use the non-volatile memory 37 as the BIOS memory.

L. Claim 13: “The method of claim 1, wherein a unique key is stored in a first non-volatile memory area of the computer.”

229. It is my opinion that Hellman discloses this feature. As discussed above, Hellman disclosed: the “computer” in the form of base unit 12.

230. Hellman discloses “a first non-volatile memory area” in the form of permanent memory 31. Hellman discloses that base unit 12 includes a “permanent memory 31, for example a PROM.” EX1004, 8:61-9:15. A POSA would have recognized that PROM is an acronym for programmable read-only memory, and that PROM was a non-volatile memory.

231. Hellman discloses a “unique key” in the permanent memory 31 in the form of key K stored in permanent memory 31. EX1004, 9:29-40. Hellman discloses that the key K can be the same as the secret key SK. EX1004, 9:29-40. Hellman discloses that “not two users share the same secret key.” EX1004, 9:41-45.

M. Claim 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.”

232. It is my opinion that, based on the modification of Hellman to store the authorization A instead of the plaintext value M (discussed for element 1.b), Hellman discloses this feature.

233. As discussed above, Hellman disclosed: the “agent” in the form of the update unit 36 in cooperation with authorization and billing unit 13; a “verification record, including the license record” in the form of the authorization A, as modified to Schneck to store authorization A stored at the specific memory location defined by hash value H in the non-volatile memory 37; the “program” in the form of software package 17; and the “unique key” in the form of the key K, which can be the same as the secret key SK.

234. Hellman discloses that the software package 17 includes data used to form the authorization A. Hellman discloses that the entirety of the software package 17 is used to generate the hash value H. EX1004, 6:31-61, 9:16-28. The hash value H is part of the authorization A. EX1004, 6:62-7:2. Hellman discloses that the authorization A is encrypted using the secret key SK, which can be the same as the key K. EX1004, 6:62-7:2, 9:29-40. Based at least on these disclosures, a POSA would have recognized that Hellman discloses that the “encrypting a license record data in the program using at least the unique key.”

N. Claim 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.”

235. It is my opinion that, based on the modification of Hellman to store the authorization A instead of the plaintext value M (discussed for element 1.b), this feature would have been an obvious modification of Hellman.

236. As discussed above, Hellman disclosed: the “license-record” in the form of authorization A, as modified to Schneck to store authorization A in non-volatile memory 37; an “erasable, non-volatile memory area of the BIOS” in the form of the non-volatile memory 37, as modified by Chou to store the BIOS routines; and the “unique key” in the form of the key K. A POSA would have recognized that the non-volatile memory 37, an “erasable, non-volatile memory area of the BIOS” could also be an erasable second non-volatile memory area of the BIOS.

237. As discussed above for element 1.c, when Hellman was modified by the teachings of Schneck to store authorization A in the non-volatile memory 37, a POSA would have recognized that verification would have been performed using the authorization A. A POSA would have recognized that one way in which the verification could be performed would be to decrypt the authorization A using the unique key K for the base unit 12 in order to retrieve the value M included therein. The value M could then be used as already described in Hellman. EX1004, 10:33-

54. A POSA would have recognized that, because the authorization A was encrypted, the authorization A would have to be decrypted in order to retrieve the number of authorized uses value M.

238. I have considered the impact of secondary indicia in forming my opinions. I am unaware of any secondary indicia at this time that would impact my conclusions that these claims are obvious in view of the recited art.

V. CONCLUSION

In signing this declaration, I recognize that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required, I will appear for cross-examination within the United States during the time allotted.

I hereby declare that all statements made herein are of my own knowledge and are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Executed on 8/24/2021 in Los Gatos, CA.



Andrew Wolfe, Ph.D.

APPENDIX A

Andrew Wolfe Ph.D.

20 S. Santa Cruz Ave. Suite 101
Los Gatos, CA 95030
(408) 402-5872 (office) (408) 394-1096 (mobile)
Email: awolfe@awolfe.org

Education:

Ph.D. in Computer Engineering, Carnegie Mellon University, 1992
Visiting Graduate Student, Center for Reliable Computing, Stanford University, 1988-1989
M.S. in Electrical and Computer Engineering, Carnegie Mellon University, 1987
B.S.E.E. in Electrical Engineering and Computer Science, The Johns Hopkins University, 1985

Recent Employment:

Consultant, [October 2002-present]

Wolfe Consulting

Consultant on processor technology, computer systems, consumer electronics, software, design tools, and intellectual property issues. Testifying and consulting expert for IP and other technology-related litigation matters.

Sample clients include:

AMD	Nvidia	Samsung
IBM	Motorola	HTC
SMIC	AMKOR	Huawei
Dell	Honeywell	Western Digital
Nintendo	Kingston	Sonos
Moneygram	Arraycomm	Insilica
Synaptics	Activision	Sawstop
Mysticom	P.A.R.C.	Quester Ventures

Lecturer, [September 2013-present]

Santa Clara University

Teaching graduate and undergraduate courses on embedded computing, mechatronics and computer architecture.

Chief Technical Officer, [1999-2002]; Sr. VP of Business Development, [2001-2002]; VP, Systems Integration, S3 Fellow, [1998 – 1999]; Director of Technology, S3 Fellow, [1997 - 1998]

SONIC|blue, Inc., Santa Clara, CA (formerly S3 Inc.)

Strategic Business Development:

Developed and implemented strategy to reposition S3 from PC graphics into the leading networked consumer electronics company.

- Acquired Diamond Multimedia and coordinated integration of communications, Rio digital music, and workstation graphics divisions into S3.
- Identified and negotiated acquisitions to grow digital media businesses including Empeg, ReplayTV, and Sensory Science.
- Identified and negotiated strategic investments including Comsilica, Intellon, KBGear Interactive, Entridia, DataPlay and others.
- Developed strategy for integrated graphics/core-logic products and established a joint venture with Via Technologies to design and market these products.
- Negotiated divestiture of graphics chip business to Via and the workstation graphics division to ATI.

Product Planning and Development:

- Drove roadmap development within SONICblue product divisions.
- Managed Business Development for all product lines.
- Led New Product Development and Corporate Vision processes.
- Acting co-General Manager of Rio digital music business in 2nd half of 2001. Responsible for all areas of product development, business development, and cost management.
- Managed development of the Savage/MX and Savage/IX mobile 3D graphics accelerators and Savage/NB system logic products.

Public Relations, Public Policy and Investor Relations:

- Present company products and strategy at industry events such as CES, Comdex, and Microprocessor Forum.
- Discuss new products and initiatives with the press.
- Promote issues of interest to SONICblue to industry groups and in Washington.
- Brief analysts, and investors on company progress. Participate in quarterly conference calls.

IP Management and Licensing:

- Negotiated and managed partnership agreements including a critical cross-licensing agreement with Intel.
- Renegotiated technology-licensing agreements with IBM for workstation graphics products.
- Evaluated outside technology opportunities, managed video research and development, and managed corporate IP strategy with legal staff including patent filings, cross licensing, and litigation.

Consulting Professor, [1999-2002]

Stanford University, Stanford, CA

Teaching computer architecture and microprocessor design.

Assistant Professor [1991 - 1997]

Princeton University, Princeton, NJ

Teaching and research in the Electrical Engineering department. Research in embedded computing systems, multimedia, video signal processors, compiler optimization, and high performance computer architecture. Principal investigator or project manager for ~\$6M in funded research.

Visiting Assistant Professor, [1992]

Carnegie Mellon University, Pittsburgh, PA

Research and preparation of teaching materials on advanced microprocessor designs including new superscalar and superpipelined processor architectures.

Founder and Vice President and Consultant, [1989 - 1995]

The Graphics Technology Company, Inc., Austin, TX

Founded company to develop touch-sensitive components and systems for the first generation of PDA devices and interactive public systems. Obtained financing from Gunze Corp., Osaka, Japan. Company is now part of 3M.

Senior Electrical Engineer, [1989]

ESL - TRW, Advanced Technology Division, Sunnyvale, CA

Designed the architecture for an Intel i860-based multiple-processor digital signal processing system for advanced military applications. Designed several FPGA interface chips for VME-bus systems.

Design Consultant, [1986 -1987]

Carroll Touch Division, AMP Inc., Round Rock, TX

Developed several new technologies for touch-screen systems. Designed the first ASIC produced for AMP, a mixed-signal interface chip for controlling touch-screen sensors. Developed the system electronics, system firmware, and customer utility software for numerous products including those based on the new ASIC.

Senior Design Engineer, [1983 -1985]

Touch Technology Inc., Annapolis, MD

Advisory Boards:

Director, Turtle Beach Corporation (NASDAQ:HEAR) (formerly Parametric Sound Corporation), KBGear Interactive, Inc., Comsilica, Inc., Rioport.com, various S3 subsidiaries.

Technical Advisory Boards, Ageia, Inc., Intellon, Inc., Comsilica, Inc., Entridia, Inc., Siroyan, Ltd., BOPS, Inc, Quester Venture Funds

Carnegie Mellon University Silicon Valley Advisory Board; Johns Hopkins University Tech Transfer Advisory Board

Awards:

Micro Test-of-Time Award (in recognition of one of the ten most influential papers of the first 25 years of the symposium), 2014

Business 2.0 “20 Young Executives You Need to Know”, 2002

Walter C. Johnson Prize for Teaching Excellence, 1997.

Princeton University Engineering Council Excellence in Teaching Award, Spring 1996

AT&T/Lucent Foundation Research Award, 1996.

Walter C. Johnson Prize for Teaching Excellence, 1995

IEEE Certificate of Appreciation, 1995, 2001.

AT&T Foundation Research Award, 1993.

Semiconductor Research Corporation Fellow, 1986 - 1991.

Burroughs Corporation Fellowship in Engineering, 1985 - 1986.

Professional Activities:

Program Chair: Micro-24, 1991, Hot Chips 13, 2001.

General Chair: Micro-26, 1993, Micro-33, 2000.

Associate Editor: IEEE Computer Architecture Letters; ACM Transactions in Embedded Computing Systems

Speaker at CES, WinHec, Comdex, Intel Dev. Forum, Digital Media Summit, Microprocessor Forum, etc.

Keynote speaker at Micro-34, ICME 2002

IEEE B. Ramakrishna Rau Award committee – 2012-2016

IEEE Computer Society Awards Committee – 2015

CES Awards Judge – 2016

Entrepreneurship Mentor – Draper University

Over 50 refereed publications.**Publications since January 2006:**

Wolfe, A., “Retrospective on Code Compression and a Fresh Approach to Embedded Systems”, IEEE MICRO, July/Aug. 2016, Invited paper.

Patents:

- U.S. Pat. 5,041,701 – *Edge Linearization Device for a Contact Input System*, Aug. 20, 1991.
- U.S. Pat. 5,438,168 – *Touch Panel*, Aug. 1, 1995.
- U.S. Pat. 5,736,688 – *Curvilinear Linearization Device for Touch Systems*, Apr. 7, 1998.
- U.S. Pat. 6,037,930 – *Multimodal touch sensitive peripheral device*, March 14, 2000.
- U.S. Pat. 6,408,421 – *High-speed asynchronous decoder circuit for variable-length coded data*, June 18, 2002.
- U.S. Pat. 6,865,668 – *Variable-length, high-speed, asynchronous decoder circuit*, March 8, 2005
- U.S. Pat. 7,079,133 – *Superscalar 3D Graphics Engine*, July 18, 2006
- EP 1 661 131 B1 – *PORTABLE ENTERTAINMENT APPARATUS*, Jan. 21, 2009
- U.S. Pat. 7,555,006 – *Method and system for adaptive transcoding and transrating in a video network*, June 30, 2009
- U.S. Pat. 7,996,595 – *Interrupt Arbitration for Multiprocessors*, Aug. 9, 2011
- EP 2 241 979 B1 – *Interrupt Arbitration for Multiprocessors*, Oct. 10, 2011
- U.S. Pat. 8,131,970 – *Compiler Based Cache Allocation*, March 6, 2012
- U.S. Pat. 8,180,963 – *Hierarchical read-combining local memories*, May 15, 2012
- U.S. Pat. 8,193,941 – *Snoring Treatment*, June 5, 2012
- U.S. Pat. 8,203,541 – *OLED display and sensor*, June 19, 2012
- U.S. Pat. 8,243,045 – *Touch-sensitive display device and method*, August 14, 2012
- U.S. Pat. 8,244,982 – *Allocating processor cores with cache memory associativity*, August 14, 2012
- U.S. Pat. 8,260,996 – *Interrupt Optimization for Multiprocessors*, Sept. 4, 2012
- 101185761 (KR) – *Noise Cancellation for Phone Conversation*, Sept. 19, 2012
- 101200740 (KR) – *OLED display and sensor*, November 7, 2012
- 101200741 (KR) – *Touch-sensitive display device and method*, November 7, 2012
- U.S. Pat. 8,321,614 – *Dynamic scheduling interrupt controller for multiprocessors*, Nov. 27, 2012
- U.S. Pat. 8,352,679 – *Selectively securing data and/or erasing secure data caches responsive to security compromising conditions*, Jan. 8, 2013
- U.S. Pat. 8,355,541 – *Texture Sensing*, Jan. 15, 2013
- U.S. Pat. 8,370,307 – *Cloud Data Backup Storage Manager*, Feb. 5, 2013
- U.S. Pat. 8,398,451 – *Tactile Input Interaction*, March. 19, 2013
- JP 5241032 B2 – *Routing Across Multicore Network Using Real World or Modeled Data*, April 13, 2013
- ZL201010124820.3 – *Interrupt Optimization for Multiprocessors*, April 17, 2013
- U.S. Pat. 8,428,438 – *Apparatus for Viewing Television with Pause Capability*, April 23, 2013
- JP 5266197 B2 – *Data Centers Task Mapping*, May 10, 2013
- U.S. Pat. 8,508,498 – *Direction and Force Sensing Input Device*, August 13, 2013
- U.S. Pat. 8,547,457 – *Camera Flash Mitigation*, October 1, 2013
- U.S. Pat. 8,549,339 – *Processor core communication in multi-core processor*, October 1, 2013
- 101319048 (KR) – *Camera Flash Mitigation*, October 10, 2013
- U.S. Pat. 8,628,478 – *Microphone for remote health sensing*, January 14, 2014
- 101362017 (KR) – *Thread Shift: Allocating Threads to Cores*, Feb. 5, 2014
- 101361928 (KR) – *Cache Prefill on Thread Migration*, Feb. 5, 2014
- 101361945 (KR) – *Mapping Of Computer Threads onto Heterogeneous Resources*, Feb. 5, 2014
- JP 5487307 B2 – *Mapping Of Computer Threads onto Heterogeneous Resources*, Feb. 28, 2014
- JP 5484580 B2 – *Task Scheduling Based on Financial Impact*, Feb. 28, 2014
- JP 5487306 B2 – *Cache Prefill on Thread Migration*, Feb. 28, 2014
- 101372623 (KR) – *Power Management for Processor*, March. 4, 2014
- 101373925 (KR) – *Allocating Processor Cores with Cache Memory Associativity*, March 6, 2014
- U.S. Pat. 8,676,668 – *Method for the determination of a time, location, and quantity of goods to be made available based on mapped population activity*, March 18, 2014
- U.S. Pat. 8,687,533 – *Energy Reservation in Power Limited Networks*, April 1, 2014
- 101388735 (KR) – *Routing Across Multicore Networks Using Real World or Modeled Data*, April 17, 2014
- U.S. Pat. 8,725,697 – *Cloud Data Backup Storage*, May 13, 2014
- U.S. Pat. 8,726,043 – *Securing Backing Storage Data Passed Through a Network*, May 13, 2014
- ZL201010124826.0 – *Dynamic scheduling interrupt controller for multiprocessors*, May 14, 2014
- JP 5547820 B2 – *Processor core communication in multi-core processor*, May 23, 2014
- U.S. Pat. 8,738,949 – *Power Management for Processor*, May 27, 2014

U.S. Pat. 8,751,854 – *Processor Core Clock Rate Selection*, June 10, 2014
 JP 5559891 B2 – *Thermal Management in Multi-Core Processor*, June 13, 2014
 101414033 (KR) – *Dynamic Computation Allocation*, June 25, 2014
 JP 5571184 B2 – *Dynamic Computation Allocation*, July 4, 2014
 101426341 (KR) – *Processor core communication in multi-core processor*, May 23, 2014
 U.S. Pat. 8,799,671 – *Techniques for Detecting Encrypted Data*, Aug 5, 2014
 101433485 (KR) – *Task Scheduling Based on Financial Impact*, Aug. 18, 2014
 U.S. Pat. 8,824,666 – *Noise Cancellation for Phone Conversation*, Sept. 2, 2014
 U.S. Pat. 8,836,516 – *Snoring Treatment*, Sept. 16, 2014
 U.S. Pat. 8,838,370 – *Traffic flow model to provide traffic flow information*, Sept. 16, 2014
 U.S. Pat. 8,838,797 – *Dynamic Computation Allocation*, Sept. 16, 2014
 U.S. Pat. 8,854,379 – *Routing Across Multicore Networks Using Real World or Modeled Data*, Oct. 7, 2014
 JP 5615361 B2 – *Thread Shift: Allocating Threads to Cores*, Oct. 15, 2014
 U.S. Pat. 8,866,621 – *Sudden infant death prevention clothing*, Oct. 21, 2014
 U.S. Pat. 8,881,157 – *Allocating threads to cores based on threads falling behind threads*, Nov. 4, 2014
 ZL201080024755.5 – *Camera Flash Mitigation*, Nov 5, 2014
 U.S. Pat. 8,882,677 – *Microphone for remote health sensing*, Nov. 11, 2014
 U.S. Pat. 8,924,743 – *Securing Data Cache through Encryption*, December 30, 2014
 U.S. Pat. 8,994,857 – *Camera Flash Mitigation*, March 31, 2015
 JP 5699140 B2 – *Camera Flash Mitigation*, April 8, 2015
 ZL201080035189.8 – *Thread Shift: Allocating Threads to Cores*, June 10, 2015
 ZL201180005030.6 – *Processor core communication in multi-core processor*, June 10, 2015
 U.S. Pat. 9,143,814 – *Method and system for adaptive transcoding and transrating in a video network*, Sept 22, 2015
 ZL201080035177.5 – *Mapping Of Computer Threads onto Heterogeneous Resources*, Oct. 14, 2015
 U.S. Pat. 9,178,694 – *Securing Backing Storage Data Passed Through a Network*, November 3, 2015
 U.S. Pat. 9,189,282 – *Thread-to-core mapping based on thread deadline, thread demand, and hardware characteristics data collected by a performance counter*, November 17, 2015
 U.S. Pat. 9,189,448 – *Routing image data across on-chip networks*, November 17, 2015
 U.S. Pat. 9,208,093 – *Allocation of memory space to individual processor cores*, December 8, 2015
 U.S. Pat. 9,239,994 – *Data Centers Task Mapping*, January 19, 2016
 ZL201080036611.1 – *Allocating Processor Cores with Cache Memory Associativity*, January 20, 2016
 EP2228779 B1 – *Traffic flow model to provide traffic flow information*, Jan. 27, 2016
 U.S. Pat. 9,262,628 – *Operating System Sandbox*, February 16, 2016
 GB2485682 – *Mapping Of Computer Threads onto Heterogeneous Resources*, Sept. 28, 2016
 U.S. Pat. 9,330,137 – *Cloud Data Backup Storage Manager*, May. 3, 2016
 ZL201080035185.X – *Cache Prefill on Thread Migration*, Aug. 24, 2016
 U.S. Pat. 9,519,305 – *Processor Core Clock Rate Selection*, December 13, 2016
 U.S. Pat. 9,569,270 – *Mapping thread phases onto heterogeneous cores based on execution characteristics and cache line eviction count*, February 14, 2017
 GB2485683 – *Thread Shift: Allocating Threads to Cores*, Oct. 18, 2017
 U.S. Pat. 9,852,435 – *Telemetrics based location and tracking*, December 26, 2017.
 U.S. Pat. 9,915,994 – *Power management for processor*, March 13, 2018
 U.S. Pat. 9,927,254 – *Traffic flow model to provide traffic flow information*, March 27, 2018
 EP2254048 B1 – *Thread Mapping in Multi-Core Processors*, August 29, 2018
 U.S. Pat. 10,860,432 – *Cloud Data Backup Storage Manager*, December 8, 2020

Expert testimony by deposition or at trial – April 15, 2016 - -present

Case	Venue	Case Number
INTELLECTUAL VENTURES II LLC v JP MORGAN CHASE & CO., et al.,	Southern District of NY	13 Civ. 3777
Certain Table Saws Incorporating Active Injury Mitigation Technology and Components Thereof (Sawstop v Bosch)	ITC	337-TA-965
INTER PARTES REVIEW OF U.S. PATENT NO. RE43,931 (TCL v Ericsson)	PTO	IPR2015-01602 IPR2015-01637 IPR2015-01641 IPR2015-01646 IPR2015-01674 IPR2015-01676 IPR2015-01761 IPR2015-01806
T-Mobile, USA, Inc. v. Huawei Device USA, Inc. and Huawei Technologies Co., Ltd.	W. D. Washington	14-cv-1351
AVM Technologies, LLC v. Intel Corporation and Related Matters	District of Delaware	15-33-RGA
TCL v. Ericsson, et al.	C.D. California	8:14-cv-341
Sonos Inc. v D&M Holdings Inc. d/b/a the D&M Group; D&M Holdings U.S. Inc. and Denon Electronics (USA) LLC.	District of Delaware	14-1330-RGA
Waymo v Uber et. al	N. D. CA	3:17-cv-00939
Certain Graphics Systems, Components Thereof and Consumer Products Containing the Same	ITC	337-TA-1044
Intellisoft, Ltd. v Acer	Superior Court of California for the County of Santa Clara	1-14-CV-272381
INTER PARTES REVIEW OF U.S. PATENT NO. 7,987,294 (D&M Holdings v Sonos)	PTO	IPR2017-01045
Joe Andrew Salazar v HTC Corporation	E.D. Texas	2:16-cv-010986-JRG-RSP
Hitachi Maxell, Ltd. v. ZTE Corporation and ZTE (USA), Inc.	E.D. Texas	5:16-cv-00179 5:16-cv-00178
D&M Holdings Inc. d/b/a the D&M Group; D&M Holdings U.S. Inc. v. Sonos Inc.	District of Delaware	16-00141-RGA
INTER PARTES REVIEW OF U.S. PATENT NO. 7,663,506 (Mediatek v AMD)	PTAB	IPR2017-00101 IPR2017-00102
Papst Licensing GmbH & Co. KG v. Samsung Electronics Co., Ltd. and Samsung Electronics America, Inc.	E.D. Texas	6:15-CV-1102
HTC Corporation v. Telefonaktiebolaget LM Ericsson	E.D. Texas	6: 18-cv-00243-JRG
Seven Networks, LLC v ZTE (USA) Inc and ZTE Corporation	N. D. Texas - Dallas	3:17-CV-1495
AGIS Software Development, LLC v. HTC Corporation	E. D. Texas	2:17-cv-514
Barbaro Technologies, LLC v. Niantic, Inc	N.D. CA	3:18-cv-02955-RS
Immersion Corp. v. Samsung Electronics America, Inc. et al	E.D. Texas	2:17-cv-00572
INTER PARTES REVIEW OF U.S. PATENT NO. 7,171,526 (Northstar Innovations – Micron)	PTAB	IPR2018-01004 IPR2018-01005

CISCO SYSTEMS, INC., vs. UNILOC USA, INC., UNILOC 2017 LLC and UNILOC LICENSING USA LLC	N.D. CA	3:18-cv-04991-SI
INTER PARTES REVIEW OF U.S. PATENT NO. 8,020,014 (Intel/VLSI)	PTAB	IPR2018-01661 IPR2018-01312
INTER PARTES REVIEW OF U.S. PATENT NO Patent 9,294,799 (Comcast/Rovi)	PTAB	IPR2019-00299
Inter Partes Review of U.S. Patent No 7720929 (Unified Patents v Datascape)	PTAB	IPR2019-01115
Solas OLED Ltd., v. Samsung Display Co., Ltd., Samsung Electronics Co, Ltd., and Samsung Electronics America, Inc.,	PTAB	IPR2019-01668
INTER PARTES REVIEW OF U.S. PATENT NO Patent 8,973,069 (Comcast/Rovi)	PTAB	IPR2019-01434
U.S. Patent No. 8,448,215 IPR (Comcast v Rovi)	PTAB	IPR2019-01353
U.S. PATENT NO. 8,847,898 IPR filing (Samsung v Neodron)	PTAB	IPR2020-00234
U.S. PATENT NO. 8,610,009 IPR filing (Samsung v Neodron)	PTAB	IPR2020-00225
U.S. PATENT NO. 10,365,747 IPR filing (Samsung v Neodron)	PTAB	IPR2020-00308
AGIS Software Development LLC v. Google LLC, AGIS Software Development LLC v. WAZE Mobile Limited, and AGIS Software Development LLC v. Samsung Elecs. Co., Ltd. et al	E.D. Texas	2:19-cv-00361-JRG 2:19-cv-00359-JRG 2:19-cv-00362-JRG
Inter Partes Review of U.S. Patent No 8,112,670 (Sony)	PTAB	IPR2020-00726
U.S. Patent 8,819,505 IPR (Intel v PACT)	PTAB	IPR2020-00525
Inter Partes Review of U.S. Patent No 8,078,540 (Sony)	PTAB	IPR2020-00922
Inter Partes Review of U.S. Patent No 9,037,807 (Intel v PACT)	PTAB	IPR2020-00540
HONG KONG UCLOUDLINK NETWORK TECHNOLOGY LIMITED AND UCLOUDLINK (AMERICA), LTD., vs. SIMO HOLDINGS INC. AND SKYROAM, INC.,	N.D. CA	3:18-cv-05031-EMC
Joe Andrew Salazar v AT&T Mobility, LLC et al.	E.D. Texas	2:20-cv-00004-JRG