IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
AUSTIN DIVISION

ANCORA TECHNOLOGIES, INC., Plaintiff, v. LG ELECTRONICS INC. and LG ELECTRONICS U.S.A., INC., Defendants.	CIVIL ACTION NO. 1:20-CV-0034-ADA JURY TRIAL DEMANDED
ANCORA TECHNOLOGIES, INC., Plaintiff, v. SAMSUNG ELECTRONICS CO., LTD. and SAMSUNG ELECTRONICS AMERICA, INC., Defendants.	CIVIL ACTION NO. 1:20-CV-0034-ADA JURY TRIAL DEMANDED

EXPERT REPORT OF SUZANNE BARBER REGARDING INVALIDITY OF U.S. PATENT NO. 6,411,941

TABLE OF CONTENTS

Page

I. INTRODUCTION 1

II. BACKGROUND AND QUALIFICATIONS 2

III. DOCUMENTS AND OTHER MATERIALS RELIED UPON 5

IV. RELEVANT PATENT LAW AND LEGAL STANDARDS 6

- A. Date of Invention 6
- B. Anticipation 7
- C. Obviousness 7
- D. Indefiniteness 10
- E. Standard of Proof 10
- F. Qualification as Prior Art 10

V. CLAIM CONSTRUCTION 14

VI. RELEVANT FIELD AND LEVEL OF ORDINARY SKILL IN THE ART 17

VII. OVERVIEW OF THE ’941 PATENT 18

VIII. BRIEF SUMMARY OF THE ’941 PATENT PROSECUTION HISTORY 26

- A. First Office Action and Applicants’ Response 26
- B. Second Office Action and Applicants’ Response 27
- C. Third Office Action and Applicant’s Response 27
- D. Notice of Allowance 28
- E. Patent Owner’s Preliminary Response (“POPR”) in IPR2020-01184 29
- F. Litigation History 32

IX. OVERVIEW OF THE STATE OF ART AT THE TIME OF THE PURPORTED INVENTION OF THE ’941 PATENT 32

- A. Architecture of PCs at the Time the ’941 Patent Was Filed 33
- B. Computer Software and Programs at the Time the ’941 Patent Was Filed 35
- C. Software for PCs, BIOS, and BIOS Memory Areas at the Time the ’941 Patent Was Filed 36
- D. Symmetric vs. Public-Key Encryption Systems 40
- E. Cryptographic Algorithms 43

F. Unique vs. Pseudo-Unique Numbers and Keys At The Time The ’941 Patent Was Filed 47

G. Problems and Prior Art Solutions 52

X. THE ’941 PATENT IS NOT ENTITLED TO A PRIORITY DATE EARLIER THAN OCTOBER 2, 1997 53

XI. OVERVIEW OF THE ARBAUGH PATENT 56

XII. THE ARBAUGH PATENT IS PRIOR ART 61

A. The Arbaugh Patent Is, On its Face, Prior Art Under 35 U.S.C. § 102(a)........ 61

B. The Arbaugh Provisional Shows the Arbaugh Patent Qualifies as Prior Art to the ’941 Patent at Least as Early as October 2, 1997. 61

C. Even Assuming the ’941 Patent Were Entitled to a Priority Date Earlier Than October 2, 1997, Arbaugh Conceived of His Patent Before March 1997 72

XIII. THE ARBAUGH PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE ’941 PATENT 78

A. Claim 1 is Anticipated or Rendered Obvious by the Arbaugh Patent 79

B. Claim 2 is Anticipated by the Arbaugh Patent........ 104

C. Claim 3 is Anticipated by the Arbaugh Patent........ 109

D. Claim 6 is Anticipated by the Arbaugh Patent........ 121

E. Claim 7 is Anticipated or Rendered Obvious by the Arbaugh Patent 122

F. Claim 8 is Anticipated by the Arbaugh Patent........ 128

G. Claim 9 is Anticipated by the Arbaugh Patent........ 129

H. Claim 10 is Anticipated by the Arbaugh Patent........ 133

I. Claim 11 is Anticipated by the Arbaugh Patent........ 135

J. Claim 12 is Anticipated or Rendered Obvious by the Arbaugh Patent 136

K. Claim 13 is Anticipated by the Arbaugh Patent........ 138

L. Claim 14 is Anticipated by the Arbaugh Patent........ 139

M. Claim 16 is Anticipated by the Arbaugh Patent........ 141

XIV. OVERVIEW OF THE JABLON PATENT........ 143

XV. THE JABLON PATENT IS PRIOR ART........ 147

XVI. THE JABLON PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE ’941 PATENT 147

A. Claim 1 is Anticipated by the Jablon Patent 148

B. Claim 2 is Anticipated by the Jablon Patent 164

C. Claim 3 is Anticipated or Rendered Obvious by the Jablon Patent 166

D. Claim 6 is Anticipated by the Jablon Patent 176

E. Claim 7 is Anticipated by the Jablon Patent 178

F. Claim 8 is Anticipated by the Jablon Patent 180

G. Claim 9 is Anticipated by the Jablon Patent 182

H. Claim 10 is Anticipated by the Jablon Patent 185

I. Claim 11 is Anticipated by the Jablon Patent 186

J. Claim 12 is Anticipated by the Jablon Patent 186

K. Claim 13 is Anticipated by the Jablon Patent 187

L. Claim 14 is Anticipated by the Jablon Patent 187

M. Claim 16 is Anticipated by the Jablon Patent 189

XVII. OVERVIEW OF THE CHOU PATENT (U.S. PATENT NO. 5,892,906) 191

XVIII. THE CHOU PATENT IS PRIOR ART 192

XIX. THE COMBINATION OF THE ARBAUGH PATENT AND THE CHOU PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE ’941 PATENT 192

XX. THE COMBINATION OF THE JABLON PATENT AND THE CHOU PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE ’941 PATENT 195

XXI. OVERVIEW OF THE MIROV PATENT (U.S. PATENT NO. 6,138,236) 197

XXII. THE MIROV PATENT IS PRIOR ART 198

XXIII. THE COMBINATION OF THE ARBAUGH PATENT AND THE MIROV PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE ’941 PATENT 198

XXIV. THE COMBINATION OF THE JABLON PATENT AND THE MIROV PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE ’941 PATENT 200

XXV. OVERVIEW OF U.S. PATENT NO. 6,153,835 TO SCHWARTZ 202

XXVI. OVERVIEW OF “USING SECURE COPROCESSORS,” B. YEE, SCHOOL OF COMPUTER SCIENCE, CARNEGIE MELLON UNIVERSITY, 1994 204

A. Claim 1 is Rendered Obvious by the Combination of Schwartz and Yee 207

B. Claim 2 Is Rendered Obvious by the Combination of Schwartz and Yee 219

C. Claim 6 Is Rendered Obvious by the Combination of Schwartz and Yee 220

D. Claim 7 Is Rendered Obvious by the Combination of Schwartz and Yee 221

E. Claim 8 Is Rendered Obvious by the Combination of Schwartz and Yee 221

F. Claim 9 Is Rendered Obvious by the Combination of Schwartz and Yee 222
G. Claim 10 Is Rendered Obvious by the Combination of Schwartz and Yee 223
H. Claim 11 Is Rendered Obvious by the Combination of Schwartz and Yee 224
I. Claim 12 Is Rendered Obvious by the Combination of Schwartz and Yee 224
J. Claim 13 Is Rendered Obvious by the Combination of Schwartz and Yee 225
K. Claim 14 Is Rendered Obvious by the Combination of Schwartz and Yee 226
L. Claim 16 Is Rendered Obvious by the Combination of Schwartz and Yee 226
M. Motivation to Combine Schwartz and Yee and/or POSITA knowledge 227
XXVII. OVERVIEW OF THE CHRISTESON PATENT 239
XXVIII. OVERVIEW OF THE HELLMAN PATENT 240
XXIX. HELLMAN IN COMBINATION WITH CHOU OR CHRISTESON INVALIDATES CLAIMS 1, 2, 6, 7, AND 11-13 OF THE ’941 PATENT AS OBVIOUS 246
A. Claim 1 Is Rendered Obvious by Hellman in Combination with Chou or Christeson 246
B. Claim 2 Is Rendered Obvious by Hellman in Combination with Chou or Christeson 258
C. Claim 6 Is Rendered Obvious by Hellman in Combination with Chou or Christeson 259
D. Claim 7 Is Rendered Obvious by Hellman in Combination with Chou or Christeson 260
E. Claim 11 Is Rendered Obvious by Hellman in Combination with Chou or Christeson 261
F. Claim 12 Is Rendered Obvious by Hellman in Combination with Chou or Christeson 261
G. Claim 13 Is Rendered Obvious by Hellman in Combination with Chou or Christeson 262
XXX. SECONDARY CONSIDERATIONS 262
XXXI. INCREMENTAL VALUE OF THE ALLEGED INVENTION OF THE ’941 PATENT 266
XXXII. CONCLUSION.. 271

I. INTRODUCTION

1. My name is Suzanne Barber. I have been retained as a technical expert by Morgan, Lewis & Bockius LLP on behalf of Defendants LG Electronics Inc. and LG Electronics U.S.A., Inc. (collectively, "LGE") to provide my opinions regarding the invalidity of claims 1-3, 6-14, and 16 (the "Asserted Claims") of U.S. Patent No. 6,411,941 ("the '941 Patent").

2. Specifically, I was requested to consider issues regarding invalidity of the Asserted Claims and specifically address the following topics: the level of skill of persons who would have worked in the field around the time of the alleged invention; and if the claims are invalid as anticipated or obvious based on one or more prior art references.

3. Regarding whether the claims are invalid as anticipated or obvious, I have been asked to compare the subject matter recited for the Asserted Claims to publications, systems, and patents that qualify as prior art to the '941 Patent. I have been asked to express my opinion on the differences, if any, between the subject matter recited in each of those claims and each of the foregoing items. To the extent I conclude there are any differences, I have also been asked to express my opinion on whether the subject matter recited in each of those claims would have been obvious to a person of ordinary skill in the art in light of the technical information available to such a person at the time the patent application was filed.

4. I reserve the right to modify or supplement my opinions, as well as the basis for my opinions, based on the nature and content of the documentation, data, proof and other evidence or testimony that the Court, the Plaintiff or its experts may present or based on any additional discovery or other information provided to me or found by me in this matter.

5. My opinions in this regard are set forth in this Report and in the accompanying Appendices.

6. I am being compensated for my time at the rate of $750.00 per hour. This compensation is not contingent upon my performance, the outcome of this matter, or any issues involved in or related to this matter.

II. BACKGROUND AND QUALIFICATIONS

7. I have extensive experience in the field of cybersecurity, artificial intelligence, information security, privacy, digital identity, digital trust, and software engineering.

8. I am currently the AT&T Endowed Professor in Electrical and Computer Engineering at The University of Texas, Founding Director of the Center for Identity at The University of Texas, and Director of the Master's Degree Program in Information Security and Privacy at The University of Texas.

9. I have over 35 years of experience working in the software industry that began during my work at the Robotics Institute at Carnegie Mellon University in Pittsburgh, PA and has been carried through my research as a Professor at The University of Texas and in my role as a Senior Advisor to the U.S. Department of Homeland Security. My experience relevant to this case includes my work in software engineering, digital identity, cybersecurity, digital trust and information security and privacy.

10. I graduated with honors in 1985 with a B.S. degree in Engineering Science from Trinity University in San Antonio, TX. After working at the Robotics Institute at Carnegie Mellon in Pittsburgh, PA, I began my Master's degree in Electrical Engineering at The University of Texas at Arlington where I graduated with honors in 1988. I continued my education and research to uniquely combine software engineering and artificial intelligence to complete my PhD degree at The University of Texas at Arlington in 1992.

11. I began my career as a Research Associate at The Robotics Institute at Carnegie Mellon in Pittsburgh PA where I specialized in artificial intelligence and software engineering to

build "smart" robotic systems. In 1986, I began work at the Automation and Robotics Research Institute (ARRI) in Fort Worth, Texas as a Faculty Associate where I combined expertise in software engineering, artificial intelligence, and software security for research sponsors such as the U.S. Air Force, National Institute of Standards and Technology (NIST). SEMATECH, and the National Science Foundation. While highly unusual for a top tier university to hire within its own University System, I was fortunate to be hired as an Assistant Tenure Track Professor in the Electrical and Computer Engineering Department at The University of Texas in Austin in September 1992. At this time, I launched the Laboratory for Intelligent Processes and Systems focusing on agent-based artificial intelligence systems for a wide range of applications including cybersecurity, manufacturing, unmanned aerial vehicles, healthcare, emergency response management, epidemiological surveillance, maritime domain awareness, and others.

12. In 1995, I co-founded the Executive Master's degree program in Software Engineering for working professionals with the degree granted by the Electrical and Computer Engineering Department at The University of Texas. Professionals from over 200 companies received their M.S. degrees from this M.S. degree program while I was Director and responsible for the academic curriculum, scheduling, fiscal planning, budget oversight, and student academic counseling. I also taught Requirements Engineering and Software Architecture classes in this Executive Software Engineering M.S. degree program.

13. In September of 1997, I was granted tenure and promoted to the rank of Associate Professor in Electrical and Computer Engineering at The University of Texas. In September 2002, I was promoted to the rank of Professor and awarded the honor of the AT&T Endowed Professorship. Between 2004 and 2010, I served as the Director of Software Engineering at The University of Texas where I was responsible for strategic planning for educational and research

programs, faculty and student recruiting and mentorship. I also served as Director of Software Engineering Research Center at The University of Texas.

14. To address the significant challenges posed by the use and abuse of sensitive data identifying people, devices and organizations, I founded the Center for Identity at The University of Texas (UT CID) in 2010. The Center for Identity serves as a research center of excellence empowering individuals and organizations to make well-informed and intentional decisions with regard to the personal data they collect, use, share, and protect with the aim of increasing trust, convenience security and privacy. The Center is a public-private partnership bringing together partners from corporations, government, and academia. As the founding and current Director of the UT CID, I am responsible for strategic vision, leading research projects, and building partnerships with faculty, corporate, and government leaders.

15. In 2015, I founded the Executive M.S. degree program in Information Security and Privacy (MSISP) for working professionals to provide a multi-disciplinary curriculum encompassing technology, policy, legal, business and social science disciplines. The MSISP degree is granted by the School of Information at The University, ranked in the Top 5 in the Nation. As the MSISP Director, I am responsible for the strategic vision, academic curriculum, scheduling, fiscal planning, budget oversight, and student academic counseling. In 2019, I was appointed by the Department of Homeland Security as a Senior Advisor to offer expertise and guidance related to identity management, biometrics and software engineering. During my time at The University of Texas from 1992 to present, I have taught Introduction to Electrical and Computer Engineering, Control Theory, Manufacturing Systems, Requirements Engineering, Software Architectures, and Information Security and Privacy. My research has been sponsored by Verizon, U.S. Army, National Science Foundation, Office of Naval Research, State of Texas,

U.S. Congress, Central Intelligence Agency, Defense Threat Reduction Agency (DTRA), Department of Defense, Defense Advanced Research Projects Agency (DARPA), Naval Surface Warfare Agency, TransUnion, Schlumberger and others. My research has resulted in publications in the fields of cybersecurity, artificial intelligence, information security, privacy, digital identity, digital trust, and software engineering in numerous refereed journals, book chapters and refereed conference proceedings. My research advances at the Center for Identity have also been featured in popular media articles in Forbes, Austin American Statesman, Dallas Morning News, Austin Business Journal, San Antonio Express News, Texas CEO Monthly, Credit Union Times, Christian Science Monitor, CBS Austin, CNBC.com, Wall Street Journal, and others.

16. I am a Senior member of the Institute of Electrical and Electronics Engineers (IEEE), and member of the Advanced Computing Machinery (ACM), American Association for Artificial Intelligence, American Society of Engineering Education, and Society of Women Engineers.

17. I am a lifetime member of the Sigma Xi and Phi Kappa Phi honor societies.

18. I have submitted declarations in conjunction with IPR, Markman, and litigation. I have also been retained as a technical expert in other matters subject to confidentiality agreements, unrelated to any of the parties named in this matter.

19. Further details on my education and work experience, including a list of all publications authored by me in the previous ten years and a list of all other cases in which I have previously given testimony in the past four years are contained in my curriculum vitae (CV), which is attached to this Report as Appendix A.

III. DOCUMENTS AND OTHER MATERIALS RELIED UPON

20. In forming the opinions set forth in this Report, I have reviewed the documents

listed Appendix B, in addition to the documents cited in this Report and Appendices that are not listed there.

21. Additionally, I have utilized my own experience and expertise, including that regarding the knowledge and capabilities of a person of ordinary skill in the relevant art in the timeframe of the claimed priority date of the ’941 Patent.

22. Further, I am personally and professionally aware of Dr. Scott Nettles, including by way of having worked with him at the University of Texas. In my opinion, he qualifies as an expert in the field of network, computer, and software engineering and is knowledgeable of the level of skill in the art. I have reviewed the Invalidity Expert Report of Dr. Scott Nettles and the materials cited therein. Where I agree with his reasoning and conclusions, I have incorporated some or all of his analysis. For some discussions below, I re-confirm that my conclusions are aligned with Dr. Nettles, and in other places I have also included additional opinions as appropriate.

23. Moreover, I have reviewed the Declarations of Dr. Eres Zadok and Dr. Andrew Wolfe in Support of Petitions for Inter Partes Review of the ’941 Patent respectively filed by Samsung Electronics Co., Ltd. and TCT Mobile (US) Inc. Those Declarations address the invalidity of the ’941 Patent and some of the references discussed here. Again, where I agree with their reasoning and conclusions, I have incorporated some or all of their analysis. In other places I have also included additional opinions as appropriate.

IV. Relevant Patent Law and Legal Standards

A. Date of Invention

24. The ’941 Patent issued from U.S. Patent Application No. 09/164,777 filed October 1, 1998, and expired on October 1, 2018. The ’941 Patent purports to claim priority to Israel Application No. 124,571, filed May 21, 1998. I have been asked to use this as the earliest

priority date for purposes of my invalidity analysis. However, I note that, in its Final Infringement Contentions, Ancora claims an earliest priority date for the '941 Patent of March 31, 1997. As discussed below, it is my opinion that this date is incorrect and the '941 Patent's earliest priority date is May 21, 1998. However, even assuming the 1997 date is accurate, much of the prior art I rely upon in my opinion would predate Ancora's alleged priority date.

B. Anticipation

25. I am informed and understand that a patent claim is invalid under 35 U.S.C. § 102[1] if all elements of the claim are disclosed in a single prior art reference.

26. I understand that to establish anticipation, a prior art reference must disclose every element of the patent claim at issue, either explicitly or inherently. In addition, the anticipating reference must enable one skilled in the art to make and use the invention without undue experimentation.

27. I understand that a reference that does not expressly disclose a claim limitation may nevertheless "inherently" disclose the limitation if the matter not expressly disclosed is necessarily present in the system or method described in the reference. I also understand that the fact that a certain result or characteristic ***may*** occur or be present in the prior art is not sufficient to establish the inherency of that result or characteristic.

C. Obviousness

28. I am informed and understand that a patent claim is invalid under 35 U.S.C. § 103 if the differences between the invention and the prior art are such that the subject matter as a

[1] I have been informed and understand that the references in this Report to 35 U.S.C. §§ 102 and 103 refer to their respective pre-AIA versions. *See* America Invents Act, P.L. 112-29 (Sept. 16, 2011) at § 3(n) (amendments to 35 U.S.C. §§ 102 and 103 take effect and apply to applications that contain or contained at any time a claim with an effective filing date on or after March 16, 2013).

whole would have been obvious at the time of the invention to a POSITA to which the subject matter pertains. Obviousness, as has been explained to me, is based on (i) the scope and content of the prior art, (ii) the differences between the prior art and the claim, (iii) the level of ordinary skill in the art, and, (iv) any secondary indicia of non-obviousness (*e.g.,* "secondary considerations" such as commercial success in the market place of the claimed invention), to the extent that they exist.

29. I further understand that obviousness may be shown by showing that it would have been obvious to combine the teachings of more than one item of prior art. In determining whether a piece of prior art could have been combined with other prior art or with other information within the knowledge of a person having ordinary skill in the art, the following are examples of approaches and rationales that may be considered:

- Combining prior art elements according to known methods to yield predictable results;
- Simple substitution of one known element for another to obtain predictable results;
- Use of a known technique to improve similar devices (methods, or products) in the same way;
- Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;
- Applying a technique or approach that would have been obvious to try (choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success);
- Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to a person having ordinary skill in the art; and

- Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

30. I further understand that an invention may be obvious if one of ordinary skill in the art, facing a wide range of needs created by developments in the field, would have seen an obvious benefit to the solutions tried by the applicant. When there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, it may be obvious to a person of ordinary skill to try the known options. If a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique would have been obvious.

31. I further understand that a person of ordinary skill in the art is a hypothetical person who is presumed to be aware of all of the relevant art at the time of the invention. The person of ordinary skill is not an automaton, and may be able to fit together the teachings of multiple patents and/or printed publications employing ordinary creativity and the common sense that familiar items may have uses in another context or beyond their primary purposes.

32. I also understand that when considering the obviousness of a patent claim, one should consider whether a teaching, suggestion, or motivation to combine the references exists so as to avoid impermissibly applying hindsight when combining or modifying the prior art. I understand this test should not be applied rigidly, but that the test can be important to avoid such hindsight.

33. I also understand that one of ordinary skill in the art must have a reasonable expectation of success in combining or modifying prior art references.

34. I also understand that all elements of a claim must be considered in an obviousness analysis.

D. Indefiniteness

35. I further understand that a patent must "conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as [the] invention." 35 U.S.C. § 112, ¶ 2 (pre-AIA). I understand that the indefiniteness standard was recently articulated by the United States Supreme Court in Nautilus*, Inc. v. Biosig Instruments*, *Inc*., 134 S. Ct. 2120 (2014), which I understand held that a "patent is invalid for indefiniteness if its claims, read in light of the specification and the prosecution history, fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention." I understand that indefiniteness is to be evaluated from the perspective of someone of ordinary skill in the relevant art at the time the patent was filed, reading the claims in light of the patent's specification and prosecution history.

E. Standard of Proof

36. I understand that the standard to prove invalidity in a district court proceeding is by clear and convincing evidence. I understand that this standard is satisfied if that to be proved is highly probable or reasonably certain. This standard has been described to me as a higher standard than a preponderance of the evidence standard, but lower than beyond a reasonable doubt standard.

F. Qualification as Prior Art

37. I have been informed by counsel that a patent publication can also qualify as prior art as of the date of an earlier provisional patent application under certain circumstances that are described below:

1. Section 102 (a)

38. Section 102(a) states that:

A person shall be entitled to a patent unless . . . the invention was known or used

> by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

35 U.S.C. § 102(a) (pre-AIA).

39. I have been informed that this provision of the patent law gives the right to a patent to the first person to invent a given invention in the United States, insofar as it bars the grant of a patent to a person that files a patent after another person came up with the same invention in the United States.

2. ***Section 102 (b)***

40. Section 102(b) states that:

> A person shall be entitled to a patent unless . . . the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

35 U.S.C. § 102(b) (pre-AIA).

41. I have been informed that this provision of the patent law imposes a bar on the issuance of a patent to anyone if, one year prior to the application for patent in the United States, the invention was described in a patent or printed publication. I have been informed that this provision applies even if the patent applicant can show that they had conceived of the invention prior to the earlier publication or patent.

3. ***Section 102 (e)***

42. Section 102(e) states that:

> A person shall be entitled to a patent unless-
>
> (e) the invention was described in . . . (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for the purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

35 U.S.C. § 102(e) (pre-AIA).

43. I have been informed that this provision means that a later-granted patent can still qualify as prior art provided it was granted on "an application for patent . . . filed in the United States before the invention by the applicant for patent." I understand that this means that a reference qualifies as a prior art reference if it was filed before the effective filing date of the patent whose validity is under analysis, even if the application only matured into an issued patent later.

44. I have been informed by counsel that in determining whether a prior art patent publication is eligible to claim the priority date of an earlier provisional patent application under 35 U.S.C. § 102(e)(2), it is necessary to evaluate whether the claims of the patent publication have adequate written description support in the provisional patent application. I understand that is sometimes called a Dynamic Drinkware analysis. *Dynamic Drinkware, LLC v. National Graphics, Inc.*, 800 F.3d 1375 (Fed. Cir. 2015).

4. Section 102 (g)

45. I have been informed by counsel that, in evaluating which of two competing patent documents was invented first, and therefore which is entitled to invalidate the other, the statutory rule is set out in pre-AIA 35 U.S.C § 102(g)(2), which states that

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

35 U.S.C. § 102(g)(2) (pre-AIA).

46. I have been informed that an inventor may rely on conception occurring outside the United States only if the inventor disclosed the complete invention to a person within the

United States (unless the inventor was serving abroad on behalf of the United States, NATO, or the WTO). If an inventor conceives of an invention abroad, the foreign invention's conception date will be the date the inventor disclosed the invention within the United States.

47. I am further informed that to claim priority to the date of conception (or the date of disclosure within the United States), the inventor must work diligently from the conception (disclosure) date until the invention is reduced to practice in the United States. In other words, an inventor working abroad cannot rely on activities performed outside the United States to show reduction to practice. This means that an invention that is conceived and reduced to practice entirely outside the United States can claim priority only to the date the invention is reduced to practice again within the United States. This also means that an inventor cannot claim priority to the domestic disclosure if the foreign invention was reduced to practice entirely abroad—no matter how diligent the foreign inventor was in reducing his or her invention to practice abroad.

5. Effect of Competing Claims to Earlier Conception and Reduction to Practice of Patents

48. I have been informed by counsel that the direction to consider "the respective dates of conception and reduction to practice of the invention, but also the reasonable diligence of one who was first to conceive and last to reduce to practice, from a time prior to conception by the other," requires considering whether one of the inventors conceived of the invention first but reduced to practice only later. If the inventor was reasonably diligent during that time, then the earlier conception date may support an argument for earlier invention even if the invention was only reduced to practice later.

6. Conception and Reduction to Practice

49. I have been informed by counsel that, with respect to patents for which the application was filed before the America Invents Act came into force, an inventor may antedate

the patent's priority date to the date of conception of the patented invention. I understand that, in order to establish priority to the date of conception, it must be shown that the inventor practiced reasonably continuous diligence in reducing the invention to practice.

50. I have been further informed by counsel that an inventor seeking to antedate a pre-AIA patent application to the date of conception cannot rely on knowledge, use or activity which occurred in a foreign country, unless that inventor was serving abroad on behalf of the United States, NATO, or WTO. I understand that an inventor seeking to antedate a patent priority date must therefore demonstrate either some reduction to practice within the United States, or disclosure of the complete invention to someone within the United States. I understand that failure to demonstrate some reduction to practice within the United States, or some communication of the complete invention to someone within the United States, will preclude an inventor from predating their invention to the alleged date of conception.

V. CLAIM CONSTRUCTION

51. I have been informed by counsel that my analysis of each patent claim must utilize the construction, as provided by the Court, of every included claim term. As discussed further below, I understand that certain claim terms have been construed by the Court, so, except where otherwise noted, I have applied the Court's construction. For terms that were not subject to claim construction, I have applied the plain and ordinary meaning of the claims in the light of the experience of one of ordinary skill in this area of technology at the time the patent was filed.

52. '941 Patent Terms and Construction:

Term	Court's Construction
"using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS" (claim 1)	Plain and ordinary meaning, wherein the plain and ordinary meaning [of] "agent is "a software program or routine."

Term	Court’s Construction
“set up a verification structure” (claim 1)	Establishing or certifying the existence of a pseudo-unique key and establishing at least one license-record location” Footnote not for the jury: “Establishing at least one license-record location” may include the steps of “forming a license-record by at least partially encrypting the contents used to form a license-record with other predetermined data contents, using at least part of the pseudo-unique key; and storing the encrypted license-record.”
“license” (claim 1)	The portion of the preamble reciting “a method of restricting software operation within a license . . . .” is non-limiting, and the term “license” does not need to be construed.
“license record” (claim 1)	“Data associated with a licensed program with information for verifying that licensed program.”
“acting on the program according to the verification” (claim 1)	Plain and ordinary meaning, wherein the step of “acting on the program” may include but is not limited to “restricting the program's operation with predetermined limitations, informing the user on the unlicensed status, halting the operation of the program under question, and asking for additional user interactions.”
order of steps (claim 1)	Use of the verification structure, as described in Limitation 3, cannot complete until the “set up a verification structure” step has completed, as described in Limitation 2. “Acting on the program according to the verification,” as described in Limitation 4, cannot complete until the “verifying the program” is completed as described in Limitation 3. The “selecting a program residing in the volatile memory” as described in Limitation 1 can occur at any time. Limitation 1 = “selecting a program residing in the volatile memory” Limitation 2 = “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.”

Term	Court's Construction
	Limitation 3 = "verifying the program using at least the verification structure from the erasable non-volatile memory of the BIOS," and Limitation 4 = "acting on the program according to the verification."
"program" (claim 1)	"A set of instructions that can be executed by a computer.
"verifying the program using at least the verification structure" (claim 1)	"Confirming whether a program is licensed using at least the verification structure."
"selecting a program residing in the volatile memory" (claim 1)	Plain and ordinary meaning.
"BIOS"	Plain and ordinary meaning wherein the meaning is "An acronym for Basic Input / Output System. It is the set of essential startup operations that begin to run automatically when a computer is turned on, which test hardware, starts the operating system, and support the transfer of data among hardware devices."
"non-volatile memory" (claim 1)	"Memory whose data is maintained when the power is removed or voltage is too low."
"volatile memory" (claim 1)	Plain and ordinary meaning wherein the meaning is "memory whose data is not maintained when the power is removed." Footnote not for the jury. "For the corner case where the hard disk drive is used as virtual RAM, the data is not accessible by normal means after the power is removed."
"non-volatile memory of the BIOS" [Plaintiff] "memory of the BIOS" [Defendant] (claim 1)	This term does not require construction.
"first non-volatile memory area of the computer" (claim 7)	Plain and ordinary meaning.

53. I also understand that the parties to this litigation have agreed to the constructions

of certain terms, and that the Court adopted two of the proposed agreed constructions:

Term	Agreed Construction
"a computer including an erasable, non-volatile memory area of the BIOS of the computer, and a volatile memory area"	This portion of the preamble is limiting.
"using the key"	"Using a pseudo-unique key."

54. In my analysis, I applied the Court's constructions set forth in ¶ 52. I also applied the Agreed Constructions set forth in ¶ 53. For all remaining claim terms, I applied the plain and ordinary meaning of such terms in the context of the '941 Patent, as would have been understood by one of ordinary skill in the art at the time of the filing of the '941 Patent.

VI. RELEVANT FIELD AND LEVEL OF ORDINARY SKILL IN THE ART

55. I have considered the relevant field and level of ordinary skill in the art relevant to the '941 Patent. I understand that factors such as the education level of those working in the field, the sophistication of the technology, the types of problems encountered in the art, the prior art solutions to those problems, and the speed at which innovations are made may help establish the level of skill in the art.

56. After reviewing the '941 Patent, the relevant portions of the prosecution history of the '941 Patent, the various Patent Office Proceedings regarding the '941 Patent, Plaintiff's position regarding the field of art, and the various reference cited, it is my opinion that the relevant art of the '941 Patent is in the general fields of software licensing and software security. *See* '941 Patent, at Abstract.

57. I understand that a person of ordinary skill in the art is a hypothetical person skilled in the art, not a judge, a layperson, a person skilled in the remote arts, or a genius in the art at hand. Relevant factors in determining the level of ordinary skill in the art include the educational level of the inventor and those who work in the field. Other considerations include

various prior art approaches employed in the art, types of problems encountered in the art, the rapidity with which innovations are made, and the sophistication of the technology involved. Not all considerations may be present in every case, and the education level of inventors is not itself conclusive.

58. Based upon my knowledge of this field, I conclude that a person of ordinary skill in this art at the time of the alleged invention of the ’941 Patent (“POSITA”), and for that matter, at all subsequent times through the present, would have held at least a Bachelor’s level college degree in Computer Science, Computer Engineering or equivalent training and experience. This person would also need to have at least two years of industry experience related to software engineering, software licensing, and/or software security. Additional work or research experience can substitute for less or different education, and vice-versa.

59. I am of skill in the relevant art, of a level at least as high as ordinarily skilled persons, and I am able to provide insight and opinions into the understandings of such persons at all relevant times by virtue of my experience as an educator, researcher and inventor.

VII. OVERVIEW OF THE ’941 PATENT

60. The ’941 Patent is entitled “Method of Restricting Software Operation within a License Limitation.” The ’941 Patent is directed to preventing software piracy by verifying that a particular software program is licensed to run on a computer. ’941 Patent, 1:11-18 (“Numerous methods have been devised for the identifying and restricting of an unauthorized software program's operation. These methods have been primarily motivated by the grand proliferation of illegally copied Software, which is engulfing the marketplace. This illegal copying represents billions of dollars in lost profits to commercial software developers.”); 1:38-40 (“The present invention relates to a method of restricting software operation within a license limitation.”). More specifically, the ’941 Patent discloses that a computer must verify that a user

has a license before the computer runs a licensed program. ’941 Patent, 2:37-59.

61. Known methods at the time to prevent software piracy included software- and hardware-based methods. ’941 Patent, 1:19-32 (“Software based products have been developed to validate authorized software usage by writing a license signature onto the computer's volatile memory (*e.g.* hard disk). These products may be appropriate for restricting honest software users, but they are very vulnerable to attack at the hands of skilled system's programmers (*e.g.* ‘hackers’). These license signatures are also subject to the physical instabilities of their volatile memory media. Hardware based products have also been developed to validate authorized software usage by accessing a dongle that is coupled *e.g.* to the parallel port of the P.C. These units are expensive, inconvenient, and not particularly suitable for software that may be sold by downloading (*e.g.* over the internet)”). However, these prior-art methods were said to be “very vulnerable to attack” by hackers or “expensive, inconvenient, and not particularly suitable for software that may be sold by downloading.” *Id*.

62. In order to overcome these problems related to software and hardware-based solutions, the ’941 Patent uses elements that were all previously known to address the alleged shortcomings. For example, to restrict software operation within a license, the ’941 Patent must operate on a computer containing technology existing at the time, including a processor, a first non-volatile memory area (*e.g.*, the ROM section of the BIOS), a second non-volatile memory area (*e.g.*, the E2PROM section of the BIOS), and a volatile memory area (*e.g.*, the internal RAM memory of the computer). ’941 Patent, Fig. 1 and col. 5:10-16.

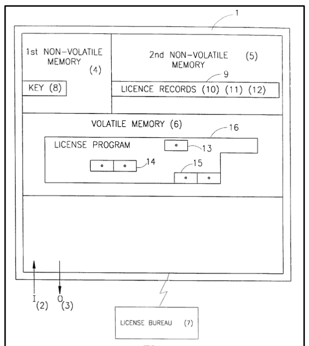

1
1st NON-VOLATILE MEMORY (4)
2nd NON-VOLATILE MEMORY (5)
9
KEY (8)
LICENCE RECORDS (10) (11) (12)
VOLATILE MEMORY (6)
16
LICENSE PROGRAM
13
14
15
I (2)
O (3)
LICENSE BUREAU (7)

FIG.1

63. As would be understood by those of skill in the art, "ROM" is a commonly understood term that stands for "Read-Only Memory." This is a non-volatile memory that can be read from, but cannot be written to or modified. Data stored in ROM is generally inserted during manufacturing and cannot be subsequently changed.

64. As would be understood by those of skill in the art, "EEPROM" (or "E^{2}PROM") is a commonly understood term that stands for "Electrically Erasable Programmable Read-Only Memory." EEPROM is a type of ROM that can be re-programmed using a special signal. Such EEPROMs can typically be erased and/or programmed when installed in a computer.

65. As would be understood by those of skill in the art, "RAM" is a commonly understood term of art that stands for "Random-Access Memory." RAM can be "accessed" to be read from and written to. The '941 patent describes two improvements that purportedly overcome the problems associated with the prior art software- and hardware-based solutions. The first improvement involves the use of a key and license record, which have been written into the non-volatile memories of a computer. '941 Patent, at 1:38-42 ("The present invention relates to a method of restricting software operation within a license limitation. This method strongly relies on the use of a key and of a record, which have been written into the non-volatile memory of a computer").

66. Here, the "key" represents a unique identification code for the computer. This key is embedded in the ROM of the computer's BIOS module. '941 Patent, at 1:45-50 ("Thus, consider a conventional computer having a conventional BIOS module in which a key was embedded at the ROM section thereof, during manufacture. The key constitutes, effectively, a unique identification code for the host computer"). Based on this disclosure, it should be understood that the key "cannot be removed or modified." 941 Patent, at 1:50-52.

67. The "license record" is a record associated with a program with information for verifying that the program is licensed, *e.g.*, the author name, program name, and a number of licensed users. '941 Patent, at 1:53-58 ("Further, according to the invention, each application program that is to be licensed to run on the specified computer, is associated with a license record; that consists of author name, program name and number of licensed users (for network). The license record may be held in either encrypted or explicit form").

68. The "key" (or some portions of the "key") is used to encrypt and/or decrypt "license record information" (or some portions of the "license record information"). '941 Patent, at 1:59-65 ("Now, there commences an initial license establishment procedure, where a verification structure is set in the BIOS so as to indicate that the specified program is licensed to run on the specified computer. This is implemented by encrypting the license record (or portion thereof) using said key (or portion thereof) exclusively or in conjunction with other identification information) as an encryption key").

69. The '941 Patent then explains that "any attempt to run a program at an unlicensed site will be immediately detected." '941 Patent, at 2:27-29. Further, as the '941 Patent teaches, if a program is licensed to run on a given computer having a first identification code (k1) stored in the ROM portion of the BIOS, the license record (LR) of the program will be encrypted

resulting in $(LR)_{k1}$ being stored in the computer's EEPROM. '941 Patent, at 2:29-35 ("Consider, for example, that a given application, say Lotus 123, is verified to run on a given computer having a first identification code (k1) stored in the ROM portion of the BIOS thereof. This obviously requires that the license record (LR) of the application after having been encrypted using k1 giving rise to $(LR)_{k1}$ is stored in the E^2PROM of the first computer").

70. If a hacker attempts to run the same program on a second computer having a second identification code (k2) stored in the ROM portion of the BIOS, even though the hacker may be able to copy the encrypted license record, $(LR)_{k1}$, to the second computer, the "hacker is unable to modify the key in the ROM of the second computer to K1, since . . . the contents of the ROM is established during manufacture and is practically invariable." '941 Patent, at 2:42-46.

71. Accordingly, when the license verifier attempts to confirm that the program is licensed to run on the second computer, it encrypts the license record (LR) using key (k2) of the second computer, which gives rise to encrypted record $(LR)_{k2}$. '941 Patent, at 2:48-53 ("Now, when the application under question is executed in the second computer, the license verifier retrieves said LR from the application and, as explained above, encrypts it using the key as retrieved from the ROM of the second computer, *i.e.* k2 giving rise to encrypted license record $(LR)_{k2}$").

72. When the second computer compares the new encrypted license record, $(LR)_{k2}$, and the copied encrypted license record, $(LR)_{k1}$, it will result in a mismatch, and the use of the application will be restricted. '941 Patent, at 2:56-59 ("It goes without saying that the data copied from the first (legitimate) computer is rendered useless, since comparing $(LR)_{k2}$ with the copied value $(LR)_{k1}$ results, of course, in mismatch").

73. With respect to the first alleged improvement, the '941 patent claims that using a

key that is unique to the computer ensures that any attempt to run the software on an unlicensed computer will be immediately detected. '941 Patent, at 2:27-59. In addition, the '941 patent teaches encrypting a license record from the program at the target computer after the software has been received by the target computer. *Id.* This is in contrast to prior art methods that encrypted the software at the software-distribution server before transmission to the target (client) computer. *See, e.g.*, U.S. Patent No. 5,892,900 ("Ginter") 137:51-56; 139:30-45.

74. The second alleged improvement disclosed in the '941 patent is the use of BIOS memory to store the encrypted license records, rather than storing such records in some other memory space. '941 Patent, at 1:59-2:5 ("Now, there commences an initial license establishment procedure, where a verification structure is set in the BIOS so as to indicate that the specified program is licensed to run on the specified computer. This is implemented by encrypting the license record (or portion thereof) using said key (or portion thereof) exclusively or in conjunction with other identification information) as an encryption key. The resulting encrypted license record is stored in another (second) non-volatile section of the BIOS, *e.g.* E^2PROM (or the ROM). It should be noted that unlike the first non-volatile section, the data in the second non-volatile memory may optionally be erased or modified (using E^2PROM manipulation commands), so as to enable to add, modify or remove licenses."); 1:43-47 ("For a better understanding of the underlying concept of the invention, there follows a specific non-limiting example. Thus, consider a conventional computer having a conventional BIOS module in which a key was embedded at the ROM section thereof, during manufacture").

75. The '941 patent notes that the memory of the BIOS is preferred because it is more secure. '941 Patent, at 3:4-9 ("[T]he 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."); '941 Patent, at 3:9-14 ("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.")

76. The claims of the '941 Patent recite a method to restrict software operation within a license by utilizing the alleged improvements noted above. As set forth in Figure 2, at a high level, the patent requires four steps:

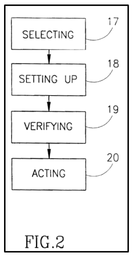


FIG.2

77. Element 17– the "selecting" step– includes the step of "establishing a licensed-software-program in the volatile memory of the computer wherein the licensed-software-program includes contents used to form a license-record." '941 Patent, at 6:7-10.

78. Element 18 – the "setting up" step – is described as "establishing or certifying the existence of a pseudo-unique key in the first non-volatile memory area; and establishing at least one license-record location in the first or the second nonvolatile memory area." '941 Patent, at 6:19-22. "Establishing a license-record includes the steps of: forming a license-record by encrypting of the contents used to form a license-record with other predetermined data contents, using the key; and establishing the encrypted license-record in one of the at least one established license record locations." '941 Patent, at 6:23-28.

79. Element 19 – the "verifying" step – is described as "encrypting the licensed-software-program's license-record contents from the volatile memory area or decrypting the license record in the first or the second non-volatile memory area, using the key; and comparing the encrypted licensed software-program's license-record contents with the encrypted license-record in the first or the second non-volatile memory area, or comparing the licensed-software-program's license-record contents with the decrypted license-record in the first or the second non-volatile memory area." '941 Patent, at 6:29-39.

80. Element 20 – the "acting" step – is described as "restricting the program's operation with predetermined limitations if the comparing yields non-unity or insufficiency." That is, restricting the program's operation if the comparison in the verifying step fails. '941 Patent, at 6:40-42.

81. The second alleged improvement involves use of the programmable portion of the BIOS memory to store the encrypted license record. '941 Patent, at 3:4-17. The patent discloses that this was important because hacking information stored in BIOS allegedly required a higher skill level than hacking information in other storage locations. *Id.* Further, hacking information on BIOS ran a higher risk of causing damage to the computer if the hack was unsuccessful. *Id.*

82. However, to overcome the known difficulties in storing the information in the BIOS memory, the '941 Patent also taught the use of something called an "agent." This "agent" allegedly allows software licensing management applications to access the BIOS memory. '941 Patent, at claim 1, 6:64-67 ("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").

VIII. BRIEF SUMMARY OF THE '941 PATENT PROSECUTION HISTORY

83. I have reviewed the prosecution history of the '941 Patent. The prosecution history does not contain any argument or discussion that I believe contradicts my opinions on invalidity.

84. The applicants filed the patent application with the U.S. Patent and Trademark Office on October 1, 1998. The originally presented claims were rejected by the examiner, and amended claims presented in two subsequent amendments were also rejected by the examiner, before any claim was allowed based on the applicants' third response submitted in March 2002.

A. First Office Action and Applicants' Response

85. All claims originally filed were rejected in light of either U.S. Patent No. 5,892,900 to Ginter et al. ("Ginter") alone, or in light of Ginter in combination with U.S. Patent No. 5,684,951 to Goldman ("Goldman"). Ginter was cited by the examiner as describing a software platform that can control the dissemination and use of electronic content. Ginter, 1:51-4:43. Ginter describes loading encryption keys into non-volatile ROM or EEPROM, acting on this encrypted data within the non-volatile memory, and selecting programs residing in volatile memory. The examiner determined that Ginter anticipated most of the pending claims. October 18, 2000, Office Action, at 2-5. The claims that were not anticipated involved the use of pseudo-unique keys and were found obvious in light of the combined teachings of Ginter and Goldman, which describes verification through use of pseudo-unique keys. *Id*. at 5-9.

86. In response, the applicants alleged that Ginter did not disclose setting up a verification structure and verifying the program using that structure. May 21, 2001 Amendment, at 6. Specifically, the applicants argued that, unlike Ginter, the claims specified a unique key already stored in the computer's non-volatile memory to generate the verification structure. The applicants amended claim 1 to require that a computer's unique key and verification structure

should be stored in the first and second non-volatile area of memory, respectively. This amendment was intended to overcome Ginter's disclosure of either distributing unique machine-bound versions of the software to the individual users' machines, or distributing the key to unlock the software with the distributed software.

B. Second Office Action and Applicants' Response

87. The examiner considered the applicants' remarks and amendment to the claims, but determined that they were not sufficient to overcome the prior art references Ginter and Goldman. The examiner repeated the previous grounds of rejection, rejecting the amended claims. June 22, 2001, Office Action at 2-12. The examiner also noted that the pending claims lacked sufficient clarity and therefore were indefinite.

88. In response, the applicants amended claim 1 to require that a verification structure be set up in an erasable, non-volatile memory of the BIOS "using an agent." November 14, 2001, Amendment at 6. However, the amendment and remarks did not explain the scope of the term "agent."

C. Third Office Action and Applicant's Response

89. In the next Office Action, the examiner rejected the pending claims, as amended, in light of U.S. Patent No. 6,189,146 to Misra et al. ("Misra") and U.S. Patent No. 5,479,639 to Ewertz et al. ("Ewertz"). January 15, 2002, Office Action at 3-6. Misra discloses a license verification structure that encrypts and stores keys in non-volatile memory, and acts on programs according to the verification structure. Ewertz teaches expanding BIOS memory to store identification and/or configuration data such as software licenses. The examiner found that Misra and Ewerts, in combination with Goldman's use of pseudo-unique keys, rendered all pending claims obvious. *Id.*

90. In response, the applicants did not amend claim 1, but sought to traverse the

rejection by arguing three points. February 5, 2002, Amendment, at 3-5.

91. First, the applicants argued that the combination of Misra and Ewertz did not disclose storing a verification structure in the BIOS of a computer (the verification structure included software license information). *Id.* Instead, the applicants argued that Misra disclosed storing (i) a client system ID that uniquely identifies the client computer, and (ii) a license ID, which is similar to the recited "verification structure," in memory other than BIOS. The applicants also argued that, although Ewertz disclosed storing a software license number in BIOS, this software license number was akin to the client system ID (i) that uniquely identifies the client computer. The applicants argued that this is not equivalent to storing a "a license ID" (*e.g.*, the "verification structure") in BIOS. Therefore, the applicants argued that the combination of Misra and Ewertz did not disclose storing the "verification structure" in BIOS.

92. Second, the applicants argued that Misra and Ewertz could not be combined. The applicants argued that Misra disclosed an operating system (OS) program whereas Ewertz disclosed a BIOS program, and that such OS and BIOS level programs could not run at the same time, *i.e.*, they could only run mutually exclusively.

93. Third, the applicants argued that using BIOS to store *application data* was not obvious. *Id.* In particular, applicants argued that a POSITA would not consider the BIOS to be a storage medium for storing application data, because (i) an operating system does not recognize BIOS memory as a storage device and cannot write to it; and (ii) there is no file system for organizing and controlling how data is stored or retrieved that is associated with BIOS.

D. Notice of Allowance

94. On March 28, 2002, the examiner issued a Notice of Allowance. In the Notice of Allowance, the examiner set out two key points in the Reasons for Allowance.

95. First, the examiner stated that prior art did not disclose an operating system level

program interacting with BIOS to verify that a program has been properly licensed using a verification structure. More specifically, the examiner stated:

> It is well known to those of ordinary skill in the art of software licensing to monitor the use of software using special code that enforces the preferences of the software provider (*e.g.* creator, distributor, or service provider), or provider and end-user, by restricting the manner in which an end-user can manipulate (*e.g.* print, save, redistribute, customize) the software. For example, Ginter et al. (US 5,892,900) implement their software distribution system by dynamically linking a verification structure, such as a PERC or permission record, to software content that dynamically control how the software, and its associated administrative data, may be distributed and used (column 155, lines 46-51). Misra et al. (US 6,189,146) disclose a method for licensing software that uses agents to manage software licenses, and stores the licenses in persistent nonvolatile storage (column 12, lines 8-31). Neither reference teaches utilizing BIOS as the non-volatile means for storing a licensed software verification structure. Ewertz et al. (US 5,479,639) teach the use of BIOS memory for storing licensing numbers. Hence, it appears initially, that to one of ordinary skill of the art, the combination of Ewertz et al. with either Ginter et al. and/or Misra et al., would render the present invention obvious. However, the key distinction between the present invention and the closest prior art, is that the Misra et al., and Ginter et al. systems and the Ewertz et al. system run at the operating system level and BIOS level, respectively. More specifically, the closest prior art systems, singly or collectively, do not teach licensed programs running at the OS level interacting with a program verification structure stored in the BIOS to verify the program using the verification structure and having a user act on the program according to the verification.

Notice of Allowance, p. 3-5.

96. Second, the examiner stated that a computer BIOS is not set up to manage a software license verification structure:

> Further, it is well known to those of ordinary skill of the art that a computer BIOS is not setup to manage a software license verification structure. The present invention overcomes this difficulty by using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS.

Id.

E. Patent Owner's Preliminary Response ("POPR") in IPR2020-01184

97. I understand that Samsung filed a petition requesting *inter partes* review of the '941 Patent on June 25, 2020, stating that claims 1-3 and 6-17 of the '941 Patent is invalid as

obvious in light of the following two combinations of references:

> Claims 1-2, 6-17 are unpatentable as obvious over U.S. Pat. No. 6,153,835 ("Schwartz") and the Ph.D. Thesis of Bennett Yee, "Using Secure Coprocessors", Carnegie-Mellon University, CMU-CS-94-149 ("Yee"), and
>
> Claims 1-3, 6-15 and 17 are unpatentable under as obvious over U.S. Pat. Nos. 5,935,243 ("Hasebe") and 5,852,736 ("Shipman").

See IPR Petition, at 9. I have reviewed the petition and Ancora's Patent Owner Preliminary Response to determine whether any statements made there influence my opinion on the application of the references used in this report.

98. In its POPR, Ancora emphasized its position regarding what the Federal Circuit and Patent and Trademark Office found innovative about the '941 Patent was

> where the license record is stored in the computer and the interaction of that memory with other memory to check for permission to run a program that is introduced into the computer. . . . More specifically: The method calls for storage of a license record in a 'verification structure' created in a portion of BIOS memory that, unlike the ROM of the BIOS, 'may be erased or modified'—for example, an Electrically Erasable Programmable Read Only Memory (E2PROM), which may be altered by 'using E2PROM manipulation commands.'

POPR at 12. Ancora underscored this by stating:

> prior to the '941 Patent, it simply was not contemplated that OS level programs could interact with the BIOS at all—much less "us[e] an agent to setup a verification structure in the erasable non- volatile memory area of the BIOS" as all challenged claims require.

Id. at 10.

99. Ancora made further arguments about what the scope of the BIOS limitations of the '941 Patent require, noting approvingly the statement in an earlier re-examination and commenting

> The fact that a program or table resides in non-volatile memory does not necessarily mean that it is part of the BIOS. It is therefore the case that a

reasonable examiner would only consider a table to be in BIOS if it were, at a minimum, created by a function residing in the BIOS. . . . Patent Owner's understanding is no different.

Id. at 13-14.

100. Ancora then relied on this to argue that a memory structure that includes part of the BIOS cannot be considered a BIOS memory, suggesting that the BIOS memory is only the part of a physical memory structure that includes the BIOS code. In reference to a figure of the Schwartz reference (right), Ancora argued that the memory structure labeled 250a cannot be considered a BIOS memory, because only part of the memory is used to contain a BIOS module.

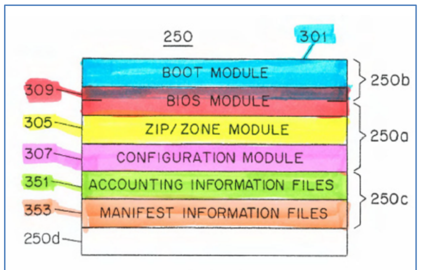


FIG. 9

POPR at 28, Figure 9, color in original. Per Ancora, the EEPROM 250a includes a portion of BIOS module 309, but also includes two other modules, labeled 305 and 307. Ancora's position appears to be that because these two modules are not called "BIOS modules," the portion of EEPROM 250a occupied by these modules cannot be considered a BIOS memory. Ancora refers to language used in the earlier ex parte reexamination that data is only considered formed "in BIOS if it were, at a minimum, created by a function residing in the BIOS." *Id.* at 30. Ancora then concludes that

> Absent a showing that the BIOS module actually uses another portion of such physical memory space as part of its normal operations, only the BIOS module would be understood by a person of ordinary skill to be "memory of the BIOS."

Id., at 30.

101. Ancora made similar arguments about the combination of Hasebe and Shipman, arguing that the references do not teach a non-volatile memory of the BIOS because the memory where data is stored is one that the BIOS facilitates access to rather than a memory that software

in the BIOS accesses directly. *Id*. at 35.]

102. I understand that the Court ruled that the terms "memory of the BIOS" and "non-volatile memory of the BIOS" need not be construed, and did not construe the terms to have the specific meaning that Ancora asserted in the POPR.

F. Litigation History

103. I understand that Ancora filed a complaint in June 2008 asserting the '941 patent against Toshiba, Dell and HP. I understand that Microsoft intervened in that case, which was ultimately dismissed in November 2009 pursuant to a settlement. However, during the case, Microsoft filed an *ex parte* reexamination request (in May 2009), which resulted in all claims being allowed.

104. I understand that Ancora filed a complaint against Apple in December 2010. As part of that litigation, the parties appealed to the Federal Circuit seeking a construction of the term "program" and whether the terms "volatile memory" and "non-volatile memory" were indefinite. Apple also filed a petition for Covered Business Method (CBM) Review in January 2016. The case was dismissed in April 2016 based on a settlement between the parties and the CBM was terminated. I understand that Ancora filed a complaint against HTC in December 2016. As part of that litigation, the District Court held that the '941 patent was invalid as it claimed ineligible subject matter. The Federal Circuit reversed in November 2018.

105. I understand that HTC filed petition for CBM Review in April 2017, which was denied. I understand that, in addition to LGE and Samsung, Ancora is asserting the '941 patent against TCL, Sony and Lenovo.

IX. OVERVIEW OF THE STATE OF ART AT THE TIME OF THE PURPORTED INVENTION OF THE '941 PATENT

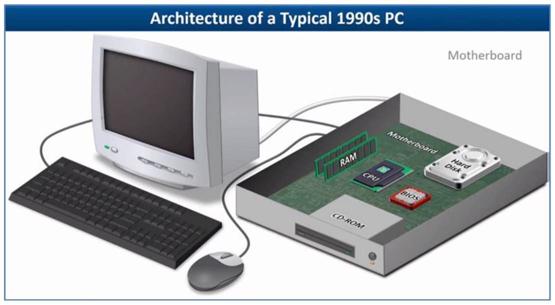
106. I was actively working in the field of software engineering and software security

at the time the ’941 Patent was filed. I am very familiar with the state of computers, software piracy, and software licensing at the time the ’941 Patent was filed.

A. Architecture of PCs at the Time the ’941 Patent Was Filed

107. The figure below depicts the architecture of a typical PC prevalent in the 1990s. The backbone of a PC is a printed circuit board that is referred to as the “motherboard.” The motherboard holds key components of a PC, including a microprocessor, Random Access Memory (or RAM), and a BIOS module. BIOS stands for Basic Input Output System. The motherboard also provides connections to a hard disk drive and other components.

F



108. The microprocessor is the brain of the computer and executes all of the instructions, meaning it runs the programs. The microprocessor may also include an identifier, such as the serial number of the microprocessor, that uniquely identifies that microprocessor or the computer that contains that microprocessor

109. When the computer is operating, the microprocessor stores programs temporarily in RAM. RAM needs a constant power supply to store data. The data stored in RAM is lost when the computer is powered down or restarted.

110. Static RAM (SRAM) and Dynamic RAM (DRAM) are just two forms of volatile Random-Access Memory (RAM) and only differ in their internal electrical circuitry. *See* Microsoft Computer Dictionary at 166–167 (“dynamic storage [] n. 1. Information storage

systems whose contents will be lost if power is removed from the system. RAM (random access memory) systems are the most common form of dynamic storage, and both dynamic RAM (DRAM) and static RAM (SRAM) are considered forms of dynamic storage. See also dynamic RAM, static RAM. Compare permanent storage."); Id., at 166 ("dynamic RAM [] n. A form of semiconductor random access memory (RAM). Dynamic RAMs store information in integrated circuits containing capacitors. Because capacitors lose their charge over time, dynamic RAM boards must include logic to refresh (recharge) the RAM chips continuously. While a dynamic RAM is being refreshed, it cannot be read by the processor; if the processor must read the RAM while it is being refreshed, one or more wait states occur. Despite being slower, dynamic RAMs are more commonly used than RAMs because their circuitry is simpler and because they can hold up to four times as much data. Acronym: DRAM (dram, D'ram). *See also* RAM. Compare static RAM.)"; Id., at at 449 ("static RAM [] n. A form of semiconductor memory (RAM) based on the logic circuit known as a flip-flop, which retains information as long as there is enough power to run the device. Static RAMs are usually reserved for use in caches. Acronym: SRAM (S'ram, S`R-A-M'). See also cache, RAM. Compare dynamic RAM.)".

111. A hard disk drive is connected to the motherboard. The hard disk provides long term storage, with or without power, for the computer operating system, software programs, and data. Such storage is "non-volatile."

112. In the 1990s, the BIOS was a separate module that stored BIOS software, which was also referred to as "firmware." The BIOS software executed during startup and was stored in a Read Only Memory (called ROM), when the computer was manufactured. Because ROM is read-only, it could not be modified. However, as discussed below, later in the 1990's, manufacturers began to store certain parts of BIOS in erasable, non-volatile memory.

113. In addition, during the manufacturing process, the PC manufacturer would often burn the computer's unique identifier, which the '941 Patent refers to as the "key," into the BIOS module. Both the key and the BIOS software were stored in ROM to provide security. It would have been nearly impossible for a hacker to modify the contents of a ROM.

B. Computer Software and Programs at the Time the '941 Patent Was Filed

114. Computer software and computer programs are typically written in a high-level human-readable language. At the time the '941 Patent was filed, software may have been written in languages such as C or C++. In order for such high-level source code to be understood by a CPU (*e.g.*, Intel or AMD processor), it would need to be "translated" using a compiler or script processor into machine level instructions understood by the CPU. Such compiled code could then be executed as a standalone program, or could be relied on by other executing code (such as a library or an included component).

115. These machine level instructions—generally called software or programs at the time—are stored in files on persistent media (*e.g.*, hard disk drive). When executed, such programs were loaded (and cached) into volatile memory (RAM), and then loaded into the CPU for execution. All computers at the time, as well as modern computers, operated in this manner. That is, all data and instructions that are executed by the CPU must be present in the CPU memory locations (called registers), or must be accessible in RAM.

116. A POSITA would understand that a CPU load programs from RAM and not directly from a hard disk or other non-volatile media. One key reason for this is because RAM is significantly faster than non-volatile media (on the order of 1,000 times faster, or more). Further, a POSITA would understand that faster memory is also more costly, so the size of RAM storage was typically much smaller than non-volatile storage.

C. Software for PCs, BIOS, and BIOS Memory Areas at the Time the ’941 Patent Was Filed

117. The software employed on PCs in the 1990s typically consisted of three layers, often referred to as a “software stack.” On the bottom was the BIOS software, above that the operating system (OS), and above that, the applications.

118. When such a computer is powered on, the microprocessor automatically runs the BIOS software from the BIOS module to initialize the computer and to verify that all the necessary components are present and operational.

119. Once the microprocessor has finished executing the BIOS software, it typically executes a software program from a particular sector on the hard disk, referred to as the “boot sector.” This program is referred to as a “boot loader” or “boot manager.” It enables the microprocessor to locate the operating system on the hard disk drive and load it into RAM. Once the operating system has been loaded in RAM, the microprocessor next executes software routines from the operating system. Once the computer has “booted” (meaning routines from the operating system have executed), a user can run applications, for example, Microsoft Word or Excel.

120. During the 1990s, several companies introduced BIOS modules that also had a programmable portion. The programmable portion was implemented using an Electrically Erasable Programmable Read-Only Memory that could be erased and reprogrammed. This is typically referred to as an “EE” PROM or “double E” PROM. In my report, I may use these terms interchangeably.

121. BIOS modules that had programmable portions were also used and disclosed in many prior art references. *See, e.g.*, *Chess* at 2:25-3:2 (discussing an alterable ROM that stores part of the bootstrap code); *Olarig* at 1:11-33, 4:16-26 (discussing updating over the network the

part of BIOS stored in non-volatile flash or EEPROM memory); *Mirov* at 1:25-35, 3:56-4:25 (discussing flash PROM that has an erasable section for storing upgradable boot-up code).

122. PC manufacturers began to use BIOS modules that had both a ROM portion and an EEPROM portion because the EEPROM portion, which is programmable, could be used to provide updates to the computer and to store data. In some instances, a small portion of the EEPROM would remain unused.

123. The EEPROM technology is not an alleged novel feature of the ’941 Patent. Rather, those of skill in the art at the time would have understood that BIOS modules containing both ROM and EEPROM existed, and were known prior to the filing of the ’941 Patent.

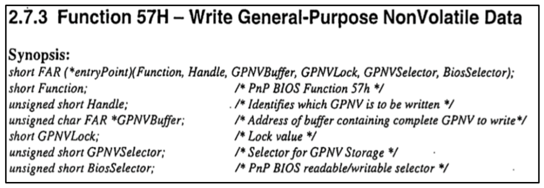
124. In the late 1990s, the operating system identified various storage devices connected to the computer and provided routines to read and write data from those devices. For example, 5.25” and 3.5” floppy drives, as well as internal hard drives, were typically used in computers of the time. The software applications, using well defined “system call” APIs, could rely upon the operating system to access the storage device in order to read and write data. These storage devices had an associated file system that would dictate where, and how, the data would be stored for efficient access. For example, at the time, the file structure would typically include a drive name, directories, files and certain metadata.

125. At the time the ’941 Patent was filed, the known BIOS modules were not accessible as a storage device by the operating system and also did not have a file system. Thus, it was significantly more difficult to access EEPROM memory locations than those on a hard drive, which operated with a known file system. To facilitate access to EEPROM sections of the BIOS modules, vendors of BIOS modules would provide a special software interface to read from, erase, and write to the programmable portion of the BIOS module. Software applications

would rely upon the software interfaces provided by the BIOS vendors to access data in the EEPROM sections of the BIOS module.

126. In 1996, a consortium of major computer and BIOS manufacturers including AMI, Dell, Intel, and Phoenix, released a second version of the "Desktop Management BIOS Specification" ("DMI Spec"). *See* DMI Spec. The DMI Spec provides a detailed specification of an API for any user or program, such as an application program running during normal execution of the operating system, to be able to access memory in a BIOS module generically, and to be able to read and write to it. One such function is "Set DMI Structure," used to write data to a writeable BIOS memory area. *Id.*, at 9 ("This function will set the DMI structure identified by the type (and possibly handle) found in the DMI structure header in the buffer pointed to by dmiDataBuffer."). The DMI Spec includes C code prototypes and machine code (written in another language called "assembly") as examples to implement each of these APIs, intended to make it easier for developers to implement and use the API. *Id.,* at 9–12. Other DMI Spec functions include reading BIOS memory. *Id*. at 8–9 ("Get DMI Structure" function). These "Set" and "Get" DMI functions are required and intended to access structured data such as BIOS configuration parameters in BIOS memory. *Id*. at 8 ("Required for DMI BIOS Support", "These structures represent the DMI information that is embedded in the System BIOS.").

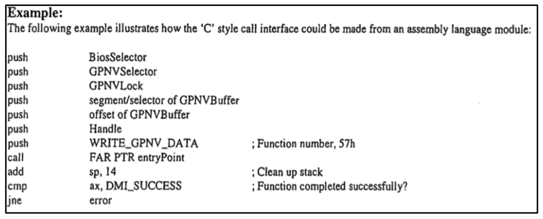
127. The DMI Spec discloses fairly similar functions to Set (write) and Get (read) more generic BIOS memory, called the general purpose non-volatile (GPNV) storage areas. *Id.* at 8–9, 17–21. *See* example C code prototype (*id.* at 20) and as well as example assembly calling code instructions *(id.* at 21) (right).



2.7.3 Function 57H – Write General-Purpose NonVolatile Data

Synopsis:

```
short FAR (*entryPoint)(Function, Handle, GPNVBuffer, GPNVLock, GPNVSelector, BiosSelector);
short Function;                     /* PnP BIOS Function 57h */
unsigned short Handle;              /* Identifies which GPNV is to be written */
unsigned char FAR *GPNVBuffer;      /* Address of buffer containing complete GPNV to write*/
short GPNVLock;                     /* Lock value */
unsigned short GPNVSelector;        /* Selector for GPNV Storage */
unsigned short BiosSelector;        /* PnP BIOS readable/writable selector */
```



Example:

The following example illustrates how the 'C' style call interface could be made from an assembly language module:

```
push        BiosSelector
push        GPNVSelector
push        GPNVLock
push        segment/selector of GPNVBuffer
push        offset of GPNVBuffer
push        Handle
push        WRITE_GPNV_DATA         ; Function number, 57h
call        FAR PTR entryPoint
add         sp, 14                  ; Clean up stack
cmp         ax, DMI_SUCCESS         ; Function completed successfully?
jne         error
```

128. These functions are considered optional, but more flexible than the one described above. *Id.* at 17 ("A General-Purpose NonVolatile (GPNV) area is a persistent general-purpose storage area managed by the Desktop Management BIOS. Multiple GPNV areas can be supported by a particular BIOS implementation. The size, format and location of a GPNV are not defined by this specification nor is the number of GPNV areas—these attributes are OEM-specific. A GPNV storage area is not a requirement for a Desktop Management BIOS. It is one method that might be used to store the System Event Log (see section 3.2.16, page 45). A GPNV storage area is not necessarily dedicated to the Desktop Management functions of the BIOS, it can also be used by other services which require non-volatile storage. A Handle parameter is passed into the GPNV function calls to specify which GPNV area is to be accessed."). A POSITA would have understood this disclosure to mean that users could store any piece of information they desired into GPNV areas (*e.g.*, system event logs, license information, encryption keys, application data, and more).

129. The API calls for the GPNV and non-GPNV are fairly similar in function: a POSITA would have understood that the DMI Spec provided different APIs to provide vendor

flexibility. The DMI Spec indicates that GPNV memory areas are full-fledged BIOS memory areas and are controlled by the BIOS software that implements the DMI specification.

130. A POSITA would have understood that "BIOS" included both software (BIOS routines to boot a computer and implement the DMI API), as well as hardware storage areas (*e.g.*, readable/writeable configuration areas and general-purpose ones). As long as the BIOS software controlled all access to read and write any BIOS memory area, a POSITA would have understood that such memory areas were an integral part of the BIOS—and it would not matter if, say, the BIOS code and its storage areas were manufactured as one electronics chip or several separate chips.

131. Operating systems include API calls, named "system calls", to open, read, and write any file. Such calls would have been well known to a POSITA and easy to use: they were taught in introductory Computer Science classes and each call included 3 arguments. The DMI Spec includes similar calls to read (Get) and write (Set) content in BIOS memory areas: these DMI Spec. calls were not as well known, not commonly taught in introductory CS courses, and their APIs were somewhat more complex (requiring 6-7 arguments each). Nevertheless, using these DMI Spec. calls were not impossible or overly onerous: a POSITA would have been able to study the DMI Spec. API and use it within a reasonable period of time.

D. Symmetric vs. Public-Key Encryption Systems

132. As further discussed below, encryption systems attempt to ensure privacy and security by protecting data. Source data, called "cleartext" or unencrypted "plaintext," is mathematically transformed using a "key", to produce output encrypted data, called "ciphertext." See *Cheswick* at 211–212. Numerous encryption algorithms, applications, and variations thereof have been developed and were well known by the late 1990s. See *Schneier* at x–xiii (part of Table of Contents).

133. The key used to convert source data (“plaintext”) into encrypted data (“ciphertext”) is generally kept a secret to ensure that the underlying data (“plaintext”) cannot be recreated (“decrypted”). The key must also be sufficiently difficult to uncover to ensure that the data is secure. That is, for an encryption algorithm to be useful, it should be strong enough to withstand attacks (*e.g.*, not selecting an easily guessed sequence like “1234” representing “ABCD”), and the key should be sufficiently long (a 4-character key is more easily guessed than a 10-character password). So long as the keys are kept safe, ciphertext can be widely and publicly disseminated, even over insecure channels, because the underlying source material cannot be decrypted. See *Cheswick*, at 211–212; *Schneier,* at 28–29.

134. One form of encryption is called symmetric, shared-key, secret-key, or private-key encryption. Here, the same key is used to encrypt and decrypt the data, hence it is symmetric. The key has to be available to both the sender that encrypts the data as well as the recipient that decrypts the data, hence the key is shared. The key must be kept secret by both sender and recipient who encrypt and decrypt the data, otherwise any hacker can decrypt the ciphertext (or encrypt the plaintext). Such encryption is known as private- or secret-key encryption. See *Cheswick* at 212–214; *Schneier* at 28–29.

135. Another form of encryption uses two keys: one called “public” and another called “private.” As expected from their names, one key is made public and the other kept private. Because of this, this scheme is known as a “public-key” or asymmetric encryption. See *Cheswick* at 217–219; *Schneier* at 31–34. For example, in this encryption scheme, the public key can be given out publicly to anyone, but the private key is kept secret. With public-key encryption, data may be encrypted using the public key, but it can only be decrypted using the private key. So, any sender can encrypt a piece of data with the public key and transmit the data

over the internet (even if it is transmitted insecurely). That is, even if a hacker has access to the encrypted data, the hacker cannot decrypt it because he or she does not have the private key. Similarly, a sender can encrypt data using the private key, and the recipient may then decrypt using the public key. In either situation, because there are two different keys, one used for encryption and a different one for decryption, this scheme depicts an asymmetric encryption algorithm.

136. Public-key cryptography has been known and used since the 1970s, popularized by Diffie and Hellman, as well as Rivest, Shamir, and Adleman (whose names serve as the foundation for the acronym RSA). See also *Cheswick* at 262, 272.

137. I note that symmetric key encryption, (also known as private-key encryption), requires the use of a“private” key for both encryption and decryption. Hence, the key is held “private” to both sides of the communication, both the sender and the receiver. Public-key cryptography (asymmetric) also uses a “private key,” but private only to one side of the communication, the recipient or sender. However, this use of a “private key” does not make such asymmetric encryption a “private key” encryption scheme.

138. The choice between using symmetric vs. public-key cryptography depends on the vendor’s or user’s needs. While symmetric encryption is faster, it is considered less secure. Conversely, public-key cryptography is considered more secure and also useful for authentication, but is much slower than symmetric encryption. In practice, many encryption systems combine both techniques to balance speed and security. For example, symmetric encryption may be used for a particular time-limited session, but that key may be transmitted using public-key cryptography). See *Schneier* at 216 (“Public-Key Cryptography Versus Symmetric Cryptography”).

E. Cryptographic Algorithms

139. The cryptographic aspects of computer security can be viewed as having two parts. First are the basic cryptographic algorithms and second are the systems and protocols built on top of those algorithms. At the heart of this is encryption, which uses a key or keys to encipher ("encrypt"") information such that it cannot be understood without decryption using a key.

1. Single/Secret Key Ciphers

140. The simplest (and oldest) ciphers use a single key to encipher ("encrypt") plaintext into ciphertext that requires that same key to decipher ("decrypt") ciphertext back into plaintext. Because there is only a single key, it must be only be known to entities that should be able to do the decryption. Hence the name secret key.

141. The classical example is the Caesar Cipher which enciphers text by rotating each character to the character that is N positions away in the alphabet. N is the key and obviously, if you know N, it is easy to decipher ciphertext into the plaintext. This cipher is easy to break since the key space (possible values of N) is small and it is not hard to try all of them.

142. Another well -known secret key system is the Data Encryption Standard (DES).

143. Here the keys are 56 independent bits and for a long-time computers were not fast enough for a brute force attack, but that has changed, leading to the development of triple DES with 168 bits in the keys or Advanced Encryption Standards with up to 256 bits.

2. Dual/Public Key Ciphers

144. A more recent invention is the idea of public key encryption (asymmetric encryption). These systems use two keys, a private one that should be secret to one node in a communication transaction (a data sender or recipient) and the public key, which is not secret and can be distributed to the public. The idea is simple, if the secret key is used to encipher the

plaintext, the public key can be used to decipher it. Similarly, if the public key is used to encipher the plaintext, the private key can be used to decipher it.

145. One advantage of public key (asymmetric encryption) systems is that it makes key distribution easier. In particular, by using public key or asymmetric encryption, keys are easier to distribute since there are few keys to manage, the private key is held private to one node not a pair of nodes (sender and receiver) and one key is public and does not need to kept secret. Ensuring keys are managed properly and not easily captured by hackers provides secrecy, one of the key goals of any such system.

146. A less intuitive use for public key (asymmetric) systems is to form a basis for authentication.

147. Asymmetric encryption relies on private/pubic key pairs – Entity A' private key corresponds to Entity A's public key. The idea of using asymmetric encryption for authentication is simple, an entity A enciphers plaintext with its private key that is secret, private only to Entity A. Now if Entity A's corresponding public key can be used to decipher the ciphertext, as long as we are sure that public key is for that entity A, we can be sure that entity A generated the ciphertext (because only Entity A's public key can decrypt data encrypted using Entity A's private key). This is an essential building block for authentication.

148. As noted above, the best-known public-key cipher is RSA (named after its inventors, Rivest, Shamir, and Adelman). The key's in RSA are based on the known hard mathematical problem of factoring the product of two large primes and as long as that problem remains hard to compute RSA will be hard to break. Part of the interest in quantum computing is that factoring primes becomes much easier with such an approach.

3. ***<u>Cryptographic Checksums/Hashes</u>***

149. Unfortunately, encryption especially public key encryption is computationally

expensive. It is also overkill when the goal is not to keep information secret, but rather to just make sure it has not been changed. This is where error codes known as message digests, hash codes, checksums, etc., come into play.

150. Checksums are used throughout computer systems to provide protection against tampering (integrity attacks). The idea is that a summary of the information is created such that if the information changes the summary is not the same.

151. A simple example would be to add up all the ones in the binary representation of the information. Now we can test if the information has been changed after transmission by adding up the ones of the information received and seeing if it has changed. Of course, tampering with the information in such a way as to keep the number of ones constant is not hard and even a natural process that flipped an even number of bits would defeat it. A somewhat better approach would be to add up all of the characters in the information, hence the name "checksum." Hashing similarly derives from the idea of combining together parts of the information is such a way as to make a "hash."

152. In practice of course we need checksums to be robust against whatever changes might be done to corrupt the information. Specifically, we do not want the hacker to be able to guess the checksum, then change/corrupt the information in a way that still matches the checksum. There is a wide class of checksums called cyclic redundancy checks (CRCs) that are good for protection against natural events, for example, CRCs can detect errors that involve flipping an even number of bits as well as many other errors that occur when transmitting or storing information. They are also cheap to compute and so their use is widespread.

153. However, Although CRCs are robust against natural processes they can be deliberately fooled by intelligent adversary. This is avoided by using cryptographic

checksums/hashes which use one-way functions that are relatively easy to compute but difficult to reverse. This means that it is hard (impossible) for an adversary to tamper with the information in such a way as to have the same hash. Examples of the use to such cryptographic hash functions include Message Digest 5 (MD5) and the Secure Hash Algorithm (SHA) family including SHA1.

4. Cryptographic Services and Protocols

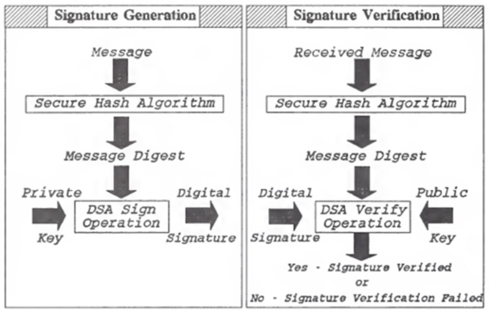
a. Digital Signatures

154. Digital signatures play a key role and are important in key distribution and protecting shared message digests. Digital signatures are attached to information and serve two purposes, first to assure that the information has not been tampered with and also to sign the information so that the entity (person) that created it can be securely identified.

155. There are a number of approaches that vary in the details, but share the same general approach. To sign a piece of information, you first generate a cryptographic hash (message digest) that once it is recovered, can be used to verify the integrity of the transmitted information. Next this hash is signed with the private key of the signer (sender). Now assuming the public key of the signer is available and is known to be the public key of the signer (part of key distribution), decrypting the signature with the public key shows that the signature was generated by the entity (otherwise the decryption would fail because only the public key of the signer could be used to decrypt hash/message digest encrypted with the signer's private key) and then checking the received information using the hash to ensure the received information was not tampered with during transmission (i.e. checking that the hash of the received information is the same as the hash sent in the signature).

156. There are a variety of implementations of this idea that are used in practice, but one that is important here is the Digital Signature Standard (DSS). DSS is a digital signature

format and has been standardized by NIST. Digital Signature Standards, Technical Report FIPS- 186, U.S. Department of Commerce, May 1994 ("DSS"). DSS uses a combination of a cryptographic hash and a private key to create a digital signature, and the public key is used to verify the signature by comparing the decoded digital hash with the cryptographic hash that is independently generated. DSS at 2.



157. Digital signatures were also described in a publication by RSA, published as "Public-Key Cryptography Standard (PKCS) #1: RSA Encryption Standard" (1993) ("PKCS1"). In RSA, then the MD5 message digest algorithm is used to generate a cryptographic hash that is then encrypted using an RSA private key. PKCS1 at 1. The processes to generate message digests and encrypt message digests to generate digital signatures are explained in PKCS1, as is the process to verify a digital signature using the public key of the signer. *Id*. at 11-15.

F. Unique vs. Pseudo-Unique Numbers and Keys At The Time The '941 Patent Was Filed

1. <u>The '941 Patent</u>

158. The '941 Patent claims "unique" and "pseudo-unique" keys. The '941 Patent describes "pseudo unique keys" in two ways.

159. First, the '941 Patent describes a pseudo-unique key as one that may "relate" to an otherwise unique identifier. *See* '941 Patent at 4:10-13 ("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."). A POSITA would have understood this to mean that a pseudo-unique key and a unique key may be interchangeable.

160. Second, the ’941 Patent describes the pseudo-unique key as one that is “probabilistically” unique. *Id*., at 4:13-18 (“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.”). A POSITA would have understood this to mean that, whatever the scheme used to generate pseudo-unique keys in this alternative embodiment, there is no 100% guarantee that any two pseudo-unique keys would not match. In other words, there is a statistical chance that two such pseudo-unique keys could be identical, and if so, a license key for one authorized computer may work on another, unauthorized computer. However, said probability is set to be extremely low that the chance of such a generation of duplicate keys is “acceptably low.” See also *Schneier* at 29–31, 44–46 (“Random and Pseudo-Random-Sequence Generation”).

2. <u>The Domain of Uniqueness</u>

161. The uniqueness property of any key is not necessarily guaranteed globally, but within a specific domain, or universe of that key. For example, with respect to U.S. social security numbers, which may be considered “keys” identifying a person, those numbers are guaranteed unique (since 1935) within the entire U.S. I note that countries outside the U.S. may assign other identifiers, *e.g*., National I.D. numbers, to identify their citizens that are unique only within that country or union. This may be called the “identity domain,” which defines the scope in which the identities (of people, devices or organizations) are defined and compared. There is no guarantee, nor a need to ensure one, that a U.S. person and a non-U.S. person would not have the same identifier. In other words, there is no guarantee, nor a need to ensure that entities in different “identity domains” would have different identifiers.

162. Here, the domain of uniqueness that’s required is a single country. Similarly, the

University of Texas assigns unique identifiers to all of its students and employees that are unique only within this university. Other universities can generate their own identifiers that are unique for them (their respective "identity domain"), but do not have to be unique across all universities. Here, the domain of uniqueness is a single university.

163. In the context of the '941 Patent, uniqueness or pseudo-uniqueness need only be guaranteed for a given vendor or entity that licenses software. For example, license keys that Microsoft issues should be unique to Microsoft's "domain," regardless of what license numbers other manufacturers, like Apple, might use to license their own products.

3. ***<u>Unique Numbers</u>***

164. At the time the '941 Patent was filed, there were generally two schemes used to attempt to guarantee uniqueness of keys: an increasing counter and a random number generator. Both of these mechanisms required storing some information persistently, *i.e.*, the "state" of the number generation. Because of this, these systems are called "stateful" schemes.

165. With respect to an increasing counter, the vendor may maintain the state of the last number ever given out. For example, if the vendor starts at 1, it can increase the number associated with the key such that each key will be guaranteed to be unique. The vendor must keep track of the last number issued (the "state") and increment it as appropriate each time a new number is needed.

166. One advantage of this scheme is that the vendor need only maintain a single integer. However, one key disadvantage is that the next numbers in the sequence may be easily predicted and generated by anyone who "breaks" the scheme used. Thus, such a simple scheme would not be implemented where privacy and security needs are high.

167. A second scheme often used by vendors is to use a large random number as a unique number. For example, a large random number could be used as a software license key or

a hardware serial number. In this scheme, the size of the random number (*i.e.*, the number of digits) is selected such that it is large enough that the vendor would not run out of random numbers for its selected domain. For example, if a vendor wishes to produce 1,000 computers with unique serial numbers, it would need to select a random number size greater than 1,000 to ensure there are no duplicates. If the vendor used a random 32-bit number, then it would have four billion possible numbers (2^{32}) for the 1,000 computers. The larger the universe of random numbers, the harder it would be for a hacker to potentially "guess" a particular key. For example, using 64-bit random numbers, the maximum number of possibilities exceeds 10^{19} (ten quintillion).

168. However, when generating random numbers, there is a mathematical possibility (however small) that two generated numbers will be identical. To ensure that duplicate numbers are not created, a vendor would need to store all previously generated numbers and consult the stored values to ensure that such a number has never been used before. This guarantees that such randomly generated numbers are 100% unique within that vendor's domain. As understood by a POSITA, where data is particularly sensitive, there would be a desire to ensure 100% uniqueness within that vendor's domain.

4. Pseudo Unique Number Schemes

169. A vendor may wish to produce many unique numbers (*e.g.*, to use as serial numbers or license keys), but may not want to maintain such a database that records all previously given numbers. In this situation, there would be no way to guarantee uniqueness because the possibility of a duplicate number exists, even if remote. Because information is not stored persistently, such a scheme is called "stateless." Here, generated numbers are no longer 100% unique, and hence called "pseudo-unique" (in the '941 Patent) or more generally "pseudo-random."

170. A vendor can reduce the probability of duplicates by (1) generating numbers in a larger space (*e.g.*, 64-bit numbers instead of 32-bit ones), and/or (2) not generating many numbers because the probability of duplicates increases the more random numbers are generated. Practically speaking, it is easier to increase the bit-length of the random numbers generated, making it large enough that a probability of duplicates is "acceptably low." See Schneier at 44–46 ("Random and Pseudo-Random-Sequence Generation").

171. Random number generation has a "collision" property, which refers to the statistical probability that two numbers will be identical or "collide." That probability is mathematically governed by the length of the random number and the number of items generated. For example, if one uses 64-bit random numbers and generates approximately 2 million numbers, there is about a 1-in-1,000,000 chance that two of the 2 million numbers will collide.

172. There are different types of random number generators. A True Random Number Generator (TRNG) uses natural processes such as the cosmos's background radiation, uranium decay, quantum effects, white noise, etc. to produce very high-quality random numbers (*i.e.*, high entropy). One disadvantage is that a TRNG is slow to produce random bits.

173. Conversely, a Pseudo Random Number Generator (PRNG) uses computer software techniques to generate random numbers. A PRNG can be thousands of times faster than a TRNG, but a PRNG's disadvantage is a higher chance of random number collisions (*i.e.*, lower entropy).

174. A vendor wishing to reduce the probability of generating duplicate random numbers (*e.g.*, to use as license keys) may choose a TRNG over a PRNG. Similarly, encryption systems can use TRNGs or PRNGs with equal ease, trading off performance for entropy levels.

175. Finally, those of skill in the art would recognize that a vendor can choose to generate pseudo-unique random numbers, and then convert them into 100% guaranteed unique numbers. Such a result would require a persistent state, *i.e.*, storing all of the generated numbers in a database and comparing any new generated numbers. A POSITA would have recognized that such a modification would be fairly easy and were well known in the art.

G. Problems and Prior Art Solutions

176. The ’941 Patent acknowledges that methods to determine whether a program on a computer is licensed and methods to securely distribute software were well known in the art before the ’941 Patent was filed. ’941 Patent, 1:11-15 (“Numerous methods have been devised for the identifying and restricting of an unauthorized software program's operation. These methods have been primarily motivated by the grand proliferation of illegally copied software, which is engulfing the marketplace”).

177. Software-based methods were available to validate authorized software usage by writing a license signature on a computer’s hard disk drive. ’941 Patent, 1:19-21 (“Software based products have been developed to validate authorized software usage by writing a license signature onto the computer's volatile memory (*e.g.* hard disk)”).

178. Hardware-based methods were available to validate authorized software usage by accessing a dongle that was plugged into the computer. ’941 Patent, 1:27-29 (“Hardware based products have also been developed to validate authorized software usage by accessing a dongle that is coupled *e.g.* to the parallel port of the P.C.”).

179. Distributing software in a secure manner was known in the prior art. Software could be distributed securely by encrypting the software prior to transmission using a key that is unique to the target computer. Here, the server encrypts the software using key k1 that is unique to PC 1 before transmitting it to PC 1. PC 1 decrypts the software using its key k1. *Yee* at 21,

Fig. 3.1. Similarly, the server encrypts the software using key k2 that identifies PC 2 before transmitting it to PC 2. PC 2 decrypts the received software using its key k2. *Id.* These transmissions are secure because the software could be decrypted only by the target PC.

180. The prior art also taught the use of the same key to encrypt the software for any computer and incorporating that key in the distributed software. *Ginter* at 137:51-56; 139:30-45.

181. The alleged invention of the '941 Patent is directed to overcoming perceived deficiencies in these prior art methods.

X. THE '941 PATENT IS NOT ENTITLED TO A PRIORITY DATE EARLIER THAN OCTOBER 2, 1997

182. I have been informed that Ancora contends that the Arbaugh Patent does not qualify as prior art because the '941 Patent is entitled to a priority date earlier than October 2, 1997. I have reviewed Dr. Nettles' analysis and concur with his opinion.

183. I understand that the '941 Patent claims a priority date of May 21, 1998 based on an earlier Israeli application, but that Ancora is asserting that the '941 Patent is entitled to a priority date no later than March 31, 1997. I understand that Ancora is asserting that Mr. Mullor and Mr. Valiko began reducing the invention to practice at least as of that date, worked diligently in reducing the invention of the '941 patent to practice, and completed the reduction to practice no later than January 5, 1998. I disagree with Ancora's theory regarding a priority date of March 31, 1997 for the reasons set forth below.

184. During his depositions in this case and in the Apple case, Mr. Mullor unambiguously stated

185. that he had resided in Israel during the period identified by Ancora as corresponding to the conception and reduction to practice of the '941 patent, and had not engaged in reduction to practice within the United States:

> Q. What was the first job that you got after you graduated from law school?
>
> A. I kept my consultancy going until we moved to this country, which was late 1998 . . .

Miki Mullor Dep Tr. (Oct. 15, 2015) at 16:7-10.

> Q. When was the last time that you conducted any activities of any form, including filing tax returns, corporation fees, et cetera, for MYPD?
>
> A. I don't think we ever filed tax returns or corporation fees or whatnot in Israel. Once, in the late 1998, I moved here, Julian stopped working on the project . . . ")

Miki Mullor Dep Tr. (Oct. 15, 2015) at 245:12-17.

> Q. [A]m I correct that all the work that was done by you or Mr. Vlaiko up through November 1998 concerning work that led to the '941 patent took place in Israel?
>
> A. Yes
>
> Miki Mullor Dep. Tr. (Nov. 2, 2020) at 26:14-18.
>
> Q. And now am I correct, up until the time you filed the application on October 1st, 1998, in the United States Patent Office, all the work that had been done by you and Mr. Vlaiko had been done in Israel?
>
> A. That is correct.

Miki Mullor Dep. Tr. (Nov. 2, 2020) at 115:23-116:3.

186. Mr. Mullor also unabiguously stated during his deposition that, to the best of his knowledge, he had not communicated the invention of the '941 patent to anyone within the United States during this time period:

> Q. Up until the time you filed the application for the U.S. Patent '941, had you disclosed the subject matter described in the '941 to anyone other than yourself and Mr. Vlaiko?
>
> A. Exclude the attorneys from that definition, no, I don't think so.

Miki Mullor Dep. Tr. (Nov. 2, 2020) at 114:9-14.

> Q. When had you first contacted lawyers in the United States with respect to the -- what became the '941 patent application?

> A. I did not contact a lawyer in the United States. My Israeli lawyer has a law firm he was working with. They made the contact.
>
> Q . And do you know when that Israeli law firm contacted the U.S. law firm?
>
> A. No.
>
> Q. Do you have any records which would indicate when that time was?
>
> A. I don't think so.

Miki Mullor Dep. Tr. (Nov. 2, 2020) at 163:10-21.

> Q. Okay. Did you tell -- did you, or to your knowledge, your coinventor tell anyone about the details of your invention other than the Israeli counsel prior to the filing of the application for the '941 patent?
>
> A Not as far as I know.

Miki Mullor Dep. Tr. (Nov. 2, 2020) at 166:15-20.

187. I understand that an inventor can antedate a patent's identified earliest priority date by demonstrating diligent reduction to practice within the United States. Mr. Mullor's testimony shows that neither he nor Mr. Valiko did any work on reducing the invention of the '941 patent to practice within the United States.

188. I understand that an inventor may also antedate their patent by showing that they communicated the complete invention to an individual within the United States. Mr. Mullor's testimony shows that he does not have any recollection or record of communicating the complete invention to anyone inside the United States prior to filing the Israeli patent application. I understand that these requirements do not apply to those working overseas on behalf of the United States, NATO or the WTO. I am not aware that Ancora has presented any evidence that Mr. Mullor was working on behalf of any of these groups during the conception and reduction to practice of the '941 patent.

189. Based on the legal principles of which counsel have informed me, this testimony indicates that Ancora cannot rely on Mr. Mullor and Mr. Valiko's conception or reduction to

practice of the invention of the ’941 patent prior to filing the Israeli patent application in order to antedate the ’941 patent. As a result, I do not believe that the ’941 patent can claim a priority date before the filing of the Israeli patent application.

190. Further, Ancora reproduced several sections of code from its ANCORA_SC_002 production. During his deposition in this litigation, however, Mr. Mullor clearly identified these code sections as dating from well after the alleged initial reduction to practice of the invention of the ’941 patent.

> Q. Okay. And is the code that's listed in the remainder of the response to interrogatory 10 code that relates to the development of your prototype prior to January 5th, 1998?
>
> A No.
>
> Q What does the code in the table indicate?
>
> A This is code I've written in 2004 as part of Ancora Online. This is parts of Ancora Online, if not a -- probably parts.

Miki Mullor Dep. Tr. (Nov. 2, 2020) at 142:22.

> Q. And am I correct that none of this code, then, was code that was developed while you were in Israel prior to November 1998?
>
> A. That is correct.

Miki Mullor Dep. Tr. (Nov. 2, 2020) at 143:14-17.

191. Since Mr. Mullor has testified that none of the code reproduced from ANCORA_SC_002 in Ancora’s Response to Interrogatory No. 10 was developed before the filing date for the ’941 patent, and - even in the situation where such code might have existed - no such code was provided to anyone in the United States - I do not believe that this code is relevant to conception or reduction to practice.

XI. OVERVIEW OF THE ARBAUGH PATENT

192. U.S. Patent No. 6,185,678 (the “Arbaugh Patent”), titled “Secure and Reliable

Bootstrap Architecture," issued on February 6, 2001 to William A. Arbaugh, David J. Farber, Angelos D. Keromytis, and Jonathan M. Smith. The Arbaugh Patent claims priority to a provisional patent application, U.S. Pat. App. No. 60/060,885, on October 2, 1997 (the "Arbaugh Provisional"). I have reviewed the Arbaugh Provisional. The Arbaugh Patent focuses on providing "integrity," that is, a reassurance that the code being executed is the code that was intended. Arbaugh Patent, Abstract, and 1:29-40; Arbaugh Provisional, 1:19-2:4. This integrity for the entire system is provided by using a "chain" of integrity checks, where each layer of the system verifies the integrity of the next layer to be executed. Arbaugh Patent, 1:32-43; Arbaugh Provisional, 1:21-2:6. The Arbaugh Patent notes that a common solution to try to avoid the issue of corruptibility of the boot process is to store relevant software within a trusted portion of hardware. Arbaugh Patent, 1:50-61; Arbaugh Provisional, 2:11-17.

193. The Arbaugh Patent discloses the "AEGIS model" that provides a solution by verifying the integrity of individual components by calculating a cryptographic hash of the program to be executed and comparing it to a cryptographic hash that is trusted to be accurate. Arbaugh Patent, 4:33-51, Arbaugh Provisional, 7:1-15. The cryptographic hash in the Arbaugh Patent is calculated using the Secure Hash Algorithm, SHA1, which I described earlier. Arbaugh Patent, 15:51-67, Arbaugh Provisional, 26:14-25. This hash is calculated using a "message," which a POSITA would have understood refers to the program binaries in the Arbaugh Patent.

194. The cryptographic hash is trusted to be accurate because it has been signed with a trusted repository's private key, which the Arbaugh Patent discloses is performed using the Digital Signature Standard, DSS. Arbaugh Patent, 15:17-49, Arbaugh Provisional, 25:17-26:13. With a DSS signature, a trusted repository will apply an asymmetric encryption technique to use one key, a private key, to generate a "signature" for a piece of data. Any other party can use the

trusted repository's public key, which like the name is publicly available, to decrypt the signature. If the decrypted signature matches the data, the other party can be assured that the signature was generated by the trusted repository.

195. The Arbaugh Patent's process uses the digital signature to authenticate the sender (by "signing" *i.e.*, encrypting with the sender's private key) and assures the integrity of the program components (using the cryptographic hash) embedded in the digital signature. If any program component's integrity check fails, the Arbaugh Patent provides that the component can be restored, or copied- over, from a "trusted repository." Arbaugh Patent, 10:13-25, Arbaugh Provisional, 15:19-16:2. This repository can either be a ROM located within the computer, or a network host that provides the necessary replacement component. Arbaugh Patent, 10:44-46, Arbaugh Provisional, 16:20-22. If the trusted repository is a ROM located within the computer, then on component failure the replacement component is simply copied over from the ROM to replace the unverified copy. Arbaugh Patent, 10:56-61, Arbaugh Provisional, 16:30-17:4. If the trusted repository is a network host, then the ROM invokes code to contact a "trusted" host that provides a signed copy of the failed component. Arbaugh Patent, 11:1-8, Arbaugh Provisional, 17:9-14.

196. The Arbaugh Patent structures its AEGIS model as a ROM that is allocated within the BIOS of the computer, as shown for example in Figure 2a:

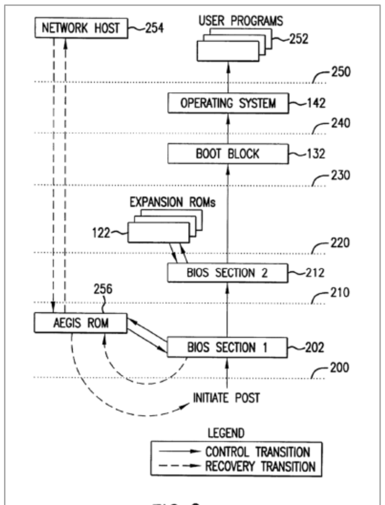
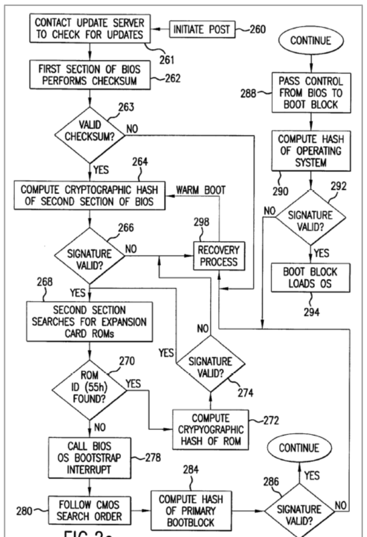


FIG. 2a

197. Arbaugh Patent, Fig. 2a; Arbaugh Provisional, Fig. 2a. Figure 2a shows BIOS section one, called "first section 202," which contains the trusted software used to implement the AEGIS model. Arbaugh Patent, 9:13-21, Arbaugh Provisional, 14:8-11. Second section 212 is a separate portion of the BIOS that includes the remainder of the system BIOS, and also has responsibility to verify the integrity of expansion ROMs. Arbaugh Patent, 9:15-16, 9:40-49, Arbaugh Provisional, 14:11-12, 14:19-14:26. The remainder of the layers above the BIOS section 2 are located outside of the BIOS, such as boot block 132 which is located on a disk medium. Arbaugh Patent, 8:23- 31, Arbaugh Provisional, 12:26-13:2. AEGIS ROM 256 is an adjunct erasable, non-volatile memory, such as a PROM, that is used either itself as the trusted repository, or as a conduit to retrieve verified code from a remote trusted repository. Arbaugh Patent, 10:47-61, Arbaugh Provisional, 16:23-17:4.

198. The Arbaugh Patent explains its main mode of operation that I rely on in my opinion as follows, by reference to the flowchart shown in Figure 2c:



CONTACT UPDATE SERVER TO CHECK FOR UPDATES
INITIATE POST
260
261
FIRST SECTION OF BIOS PERFORMS CHECKSUM
262
263
VALID CHECKSUM?
NO
YES
264
COMPUTE CRYPTOGRAPHIC HASH OF SECOND SECTION OF BIOS
WARM BOOT
266
298
SIGNATURE VALID?
NO
RECOVERY PROCESS
268
YES
SECOND SECTION SEARCHES FOR EXPANSION CARD ROMs
NO
YES
SIGNATURE VALID?
274
270
ROM ID (55h) FOUND?
YES
COMPUTE CRYPYOGRAPHIC HASH OF ROM
272
NO
CALL BIOS OS BOOTSTRAP INTERRUPT
278
284
280
FOLLOW CMOS SEARCH ORDER
COMPUTE HASH OF PRIMARY BOOTBLOCK
286
SIGNATURE VALID?
YES
NO
CONTINUE
CONTINUE
288
PASS CONTROL FROM BIOS TO BOOT BLOCK
COMPUTE HASH OF OPERATING SYSTEM
290
292
NO
SIGNATURE VALID?
YES
BOOT BLOCK LOADS OS
294

FIG.2c

199. After a computer using the Arbaugh invention undergoes a power-on self-test, it will automatically check a server for updates to its software (box 261). Following this a simple integrity check is performed on the first section of the BIOS (boxes 262-263) followed by performing a more robust verification of the second section of the BIOS (boxes 264-266). In this more robust check, a cryptographic hash is computed for the component in question (here, the second section of the BIOS) which is then compared to a trusted, digitally-signed version of the hash of the same component. This is followed by a chain of similar checks for expansion card ROMs (if present, boxes 272-274), the primary bootblock (boxes 284-286), and then performing the same check for the operating system of the computer (boxes 290-292). If any verification check in the process in Figure 2 fails, then the Arbaugh Patent invokes its recovery process (box 298), which retrieves an assumed-valid copy from the trusted repository as I described earlier. I

note that after this assumed-valid copy is retrieved from the trusted repository, the Arbaugh Patent nonetheless describes rebooting the computer and performing a full verification of the cryptographic hashes of each component again to verify as much. Id., at 10:65-11:17.

XII. THE ARBAUGH PATENT IS PRIOR ART

A. The Arbaugh Patent Is, On its Face, Prior Art Under 35 U.S.C. § 102(a).

200. The Arbaugh Patent was filed on October 2, 1998, and claims priority to a provisional patent application filed on October 2, 1997. The ’941 Patent was filed on October 1, 1998, claiming priority to an Israeli patent application filed on May 21, 1998. I understand that on the basis of these two facts, then, the Arbaugh Patent qualifies as prior art to the ’941 Patent on the basis of pre-AIA 35 U.S.C. § 102(a), provided that the Arbaugh Patent is entitled to claim priority to the October 2, 1997 Arbaugh provisional patent application.

B. The Arbaugh Provisional Shows the Arbaugh Patent Qualifies as Prior Art to the ’941 Patent at Least as Early as October 2, 1997.

201. I have compared the claims of the Arbaugh Patent to the content of the Arbaugh Provisional to confirm that the claims of the Arbaugh Patent are fully supported by the written description in the Arbaugh Provisional.

202. The Arbaugh Patent issued with seven claims, and I address each of these claims in turn, identifying the portion of the Arbaugh Provisional that discloses the claimed element of the Arbaugh Patent.

1. <u>Claim 1 of the Arbaugh Patent is Supported by the Provisional</u>

203. Claim 1 of the Arbaugh Patent recites:

> 1. An architecture for initializing a computer system comprising: a processor,
>
> an expansion bus coupled to said processor;
>
> a memory coupled to said expansion bus, said memory storing a system BIOS for execution by said processor upon power up of the computer system;

> a plurality of boot components coupled to said expansion bus and accessed by said processor when said system BIOS is executed;
>
> a trusted repository coupled to said expansion bus, and means for verifying the integrity of said boot components and said system BIOS wherein integrity failures are recovered through said trusted repository.

204. The Arbaugh Provisional discloses the "processor" in the first limitation of claim 1 of the Arbaugh Patent. The provisional states the invention "relates to an architecture for initializing a computer system and more particularly to a secure bootstrap process and automated recovery procedure." Arbaugh Provisional at 4. The computer system disclosed in the Arbaugh Provisional uses the "IBM PC architecture," id. at 13-14, and is described as including a "motherboard" and a "processor," which are both components of computer systems, and specifically IBM PC architecture computer systems, at the time to persons having ordinary skill in the art. For example, the Arbaugh Provisional discloses a "200MHz Pentium Pro" processor in an exemplary embodiment. *Id*. at 36.

205. The Arbaugh Provisional discloses the "expansion bus coupled to said processor" in the second limitation of claim 1 of the Arbaugh Patent. First, an expansion bus was a well- known standard component of IBM PC architecture computer systems at the time, as expansion buses are the way that necessary computer components like graphics cards, serial cards, and network adapters would be added to computer systems. A POSITA would have expected that an IBM PC architecture computer system at the time would need to have an expansion bus as they were generally necessary to be usable by the customers of IBM PCs. Second, the Arbaugh Provisional discloses expansion ROMs, which are described as being "ROMs on add in boards," id. at 6, and "expansion boards" that can include "expansion cards." *Id*. at 8. A POSITA would have understood that these expansion cards, boards, and ROMs would need to be connected to the processor and other components in order to work, as is

contemplated by the Arbaugh Provisional:

> Once BIOS 112 has performed all of its power on tests, it begins searching for expansion card ROMs 122, step 154, which are identified in memory by a specific signature. Once a valid signature is found by BIOS 112, step 156, control is immediately passed to the corresponding expansion card ROM 122. When expansion card ROM 122 completes its execution, step 158, control is returned to BIOS 112.

Id. at 15. A POSITA would have understood that for a ROM to be "executed," the instructions contained in that ROM need to be loaded into the computer's processor, since ROM is merely a memory that holds instructions, and those instructions can only be executed in a processor. Therefore, it would have been well-understood by a POSITA that the expansion bus in the Arbaugh Provisional would be coupled to the processor in the Arbaugh Provisional, or else the instructions in the ROMs could not be executed.

206. The Arbaugh Provisional discloses the "a memory coupled to said expansion bus, said memory storing a system BIOS for execution by said processor upon power up of the computer system" in the third limitation of claim 1 of the Arbaugh Patent. The Arbaugh Provisional describes having a system BIOS 112, see, *e.g.*, id. at 15, which it describes in the exemplary embodiment as being a "one megabit (128 kilobytes)" memory, id. at 22. A POSITA would have understood that the expansion bus in an IBM PC architecture is connected to the memory in the computer system, including the memory storing the system BIOS. The Arbaugh Provisional explicitly describes:

> First layer 110 includes system BIOS 112 and corresponds to the first phase of the bootstrap process. The first phase of the boot process is the Power on Self Test or POST. POST is invoked, step 150, in one of four ways:
>
> 1. Applying power to the computer automatically invokes POST causing the processor to jump to the entry point indicated by the processor reset vector.
>
> . . .
>
> In each of the cases above, a sequence of tests are conducted, step 152.

> All of these tests, except for the initial processor self test, are under the control of system BIOS 112.

Id. at 15. This portion of the Arbaugh Provisional supports the limitation that the system BIOS 112 is executed upon power up of the computer system. As I described earlier, a POSITA would have understood that the processor is the part of the computer that executes instructions stored in memory, and the Arbaugh Provisional therefore also discloses a memory storing the instructions for execution by the processor.

207. The Arbaugh Provisional discloses the "a plurality of boot components coupled to said expansion bus and accessed by said processor when said system BIOS is executed" in the fourth limitation of claim 1 of the Arbaugh Patent. The Arbaugh Provisional discloses a plurality of boot components coupled to the expansion bus, in the form of various components that are executed during the boot-up process. The first is in the form of expansion card ROMs that are executed as part of the boot process. *Id*. at 15 ("Once a valid signature is found by BIOS 112, step 156, control is immediately passed to the corresponding expansion card ROM 122. When expansion card ROM 122 completes its execution, step 158, control is returned to BIOS 112."). The second is in the form of bootable disks whose primary boot block is loaded following expansion card ROMs. *Id.* ("The bootstrap code first finds a bootable disk by searching the disk search order defined in the CMOS, step 162. Once a bootable disk is found, step 164, the bootstrap code loads primary boot block 132 into memory, step 166, and passes control to it, step 168."). Each one of these steps is executed during the boot process, and because they are executed, a POSITA would have understood they were executed by the processor. This execution occurs as a consequence of executing the system BIOS, and so are accessed by the processor when the system BIOS is executed, as required by this limitation.

208. The Arbaugh Provisional discloses the "a trusted repository coupled to said

expansion bus" in the fifth limitation of claim 1 of the Arbaugh Patent. The Arbaugh Provisional includes a disclosure of the trusted repository. *See, id.*, at 18 ("Any integrity failures identified in the above process are recovered through a trusted repository, step 298, as discussed below."). The trusted repository is described in detail, for example, as in Arbaugh Provisional at 18-20. One form of this trusted repository disclosed is in the form of an expansion ROM board:

> The trusted repository can either be an expansion ROM board that contains verified copies of the required software, such as AEGIS ROM 256, or it can be network host 254.

Id. A POSITA would have understood that the expansion ROM board containing the trusted repository would be coupled to the expansion bus the same as the other expansion ROM boards in the Arbaugh Provisional.

209. The Arbaugh Provisional discloses the "means for verifying the integrity of said boot components and said system BIOS wherein integrity failures are recovered through said trusted repository" in the sixth limitation of claim 1 of the Arbaugh Patent. The Arbaugh Provisional discloses verifying the integrity of boot components and the system BIOS, *see* Arbaugh Provisional, 13:8-15:18, and recovering from integrity failures using a trusted repository. Arbaugh Provisional, 15:20-17:26.

2. <u>Claim 2 of the Arbaugh Patent is Supported by the Provisional</u>

210. Claim 2 of the Arbaugh Patent discloses:

> 2. An architecture for initializing a computer system according to claim 1, wherein
>
> said trusted repository is an expansion ROM coupled to said expansion bus.

211. The Arbaugh provisional discloses the further limitation of dependent claim 2 that "said trusted repository is an expansion ROM coupled to said expansion bus." As I described for the fifth limitation of claim 1, the disclosure of the trusted repository in the Arbaugh Provisional

provides that one option for the trusted repository is as an expansion ROM board.

3. Claim 3 of the Arbaugh Patent is Supported by the Provisional

212. Claim 3 of the Arbaugh Patent discloses: 3.

An architecture for initializing a computer system according to claim 1, wherein

said trusted repository is a host computer communicating with said computer system through a communications interface coupled to said expansion bus.

213. The Arbaugh provisional discloses the further limitation of dependent claim 3 that "said trusted repository is a host computer communicating with said computer system through a communications interface coupled to said expansion bus." The alternative form of trusted repository disclosed in the Arbaugh Provisional is as a repository stored on a network host, network host 254. *Id.* at 19. The use of the trusted repository over a network is conducted as part of the AEGIS recovery protocol, which is described using well-known networking techniques such as TFTP and DHCP. *Id.* at 20, 30-32. A POSITA would have readily understood that in order for a computer to communicate with a trusted repository over a network, the computer must include a form of communications interface like a network card in order to communicate with the trusted repository. The network card would need to be coupled to the expansion bus in order to be usable by the processor, as was well understood at the time by persons having ordinary skill in the art.

4. Claim 4 of the Arbaugh Patent is Supported by the Provisional

214. Claim 4 of the Arbaugh Patent discloses:

4. A method for initializing a computer system comprising the steps of:

invoking a Power on Self Test (POST);

verifying the integrity of a system BIOS; verifving [sic] the integrity of a boot component; and when said boot component fails, recovering said failed boot.

215. The Arbaugh provisional discloses the further limitation of dependent claim 4 of

"(1) invoking a Power on Self Test (POST); (2) verifying the integrity of a system BIOS; (3) verifying the integrity of a boot component; and (4) when said boot component fails, recovering said failed boot component."[2] The Arbaugh Provisional discloses these four steps. The Arbaugh Provisional discloses a power on self-test being invoked in four different ways. *Id.* at 15 ("First layer 110 includes system BIOS 112 and corresponds to the first phase of the bootstrap process. The first phase of the boot process is the Power on Self-Test or POST. POST is invoked, step 150, in one of four ways . . ."). The Arbaugh Provisional also discloses verifying the integrity of a system BIOS, in the form of verifications of several portions of the system BIOS. For instance, the Arbaugh Provisional describes separating the system BIOS into two sections, one section of which is deemed trustworthy and has its integrity verified by a checksum. *Id.* at 17 ("First section 202 executes and performs the standard checksum calculation over its address space, step 262, to protect against ROM failures."). The following step verifies the integrity of the second section using a cryptographic hash. *Id.* ("Following successful completion of the checksum, step 263, the cryptographic hash of second section 212 is computed, step 264, and verified against a stored signature, step 266. If the signature is valid, control is passed to second section 212.").

216. After verification of the BIOS component, the Arbaugh Provisional then discloses verifying the integrity of a boot component. Specifically, the Arbaugh Provisional discloses verifying expansion ROMs:

> Once BIOS 112 has performed all of its power on tests, it begins searching for expansion card ROMs 112, step 268. Once a valid signature is found by BIOS 112, step 220, control is passed to expansion card ROM 112, step 270. However, prior to passing control to expansion ROM 112, a cryptographic hash is computed, step 272, and verified against a stored digital signature for the expansion card, step 25 274. If the signature is valid, then control is passed to the

[2] This claim language was corrected in the certificate of correction issued on November 6, 2012, which is part of and attached to the end of the copy of the Arbaugh Patent that I reviewed for this report.

> expansion ROM 112 and it is executed.

Id. at 17. Similarly, after verifying the expansion card ROMs, the Arbaugh Provisional discloses verifying the boot blocks of any bootable devices in the system:

> The bootstrap code finds a bootable device by following the CMOS search order, step 280, and verifies boot block 114, step 286, after computing a cryptographic hash of boot block 114, step 284.

Id. at 18. These both support the disclosure of verifying the integrity of boot components. Any integrity failure in this process then leads to recovering the failed boot component:

> Any integrity failures identified in the above process are recovered through a trusted repository, step 298, as discussed below.
>
> In the AEGIS boot process, either the operating system kernel is started, or a recovery process is entered to repair any integrity failure detected. Once the repair is completed, the system is restarted (warm boot) to ensure that the system boots. This entire process occurs without user intervention.

Id.

5. <u>Claim 5 of the Arbaugh Patent is Supported by the Provisional</u>

217. Claim 5 of the Arbaugh Patent discloses:

> 5. A method according to claim 4, wherein
>
> step (1) further comprises the step of performing a checksum calculation over the address space of a trusted memory location.

218. The Arbaugh provisional discloses the further limitation of dependent claim 5 of "step (1) further comprises the step of performing a checksum calculation over the address space of a trusted memory location." The Arbaugh Provisional discloses verifying the integrity of a system BIOS, in the form of verifications of several portions of the system BIOS. For instance, the Arbaugh Provisional describes separating the system BIOS into two sections, one section of which is deemed trustworthy and has its integrity verified by a checksum. *Id.* at 17 ("First section 202 executes and performs the standard checksum calculation over its address space, step

262, to protect against ROM failures.”). The memory location of the first section is deemed trusted. *Id.* (“First section 202 and AEGIS ROM 256 (if used) contain the bare essentials needed for integrity verification and recovery. They comprises [sic] the ‘trusted software’.”). A POSITA would have understood that the checksum verification is applied to the contents of the trusted memory location in order to perform the stated purpose of verifying that the contents of the ROM have not failed.

6. <u>Claim 6 of the Arbaugh Patent is Supported by the Provisional</u>

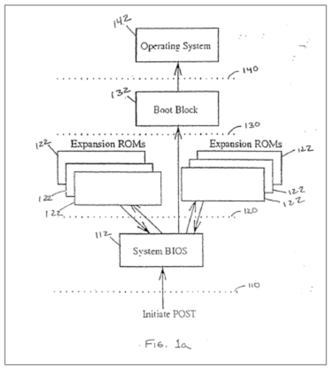
219. Claim 6 of the Arbaugh Patent discloses:

> 6. A method according to claim 4, wherein step (3) comprises the steps of:
>
> computing a cryptographic hash value for said boot component; and comparing said cryptographic hash value with a digital signature associated with said boot component stored in a trusted memory location. The Arbaugh provisional discloses the further limitation of dependent claim 6 of wherein step (3) comprises the steps of: (a) computing a cryptographic hash value for said boot component; and (b) comparing said cryptographic hash value with a digital signature associated with said boot component stored in a trusted memory location.”

220. The Arbaugh Provisional discloses both step 3(a) claimed in claim 6, of computing a cryptographic hash value for the boot component, as well as step 3(b), of comparing the cryptographic hash value with a digital signature associated with the boot component stored in a trusted memory location. As explained earlier, the step of “verifying the integrity of a boot component” is disclosed by the verification of expansion ROMs. The Arbaugh Provisional explicitly discloses these two elements, highlighted in (for example) the following passage:

> Once BIOS 112 has performed all of its power on tests, it begins searching for expansion card ROMs 112, step 268. Once a valid signature is found by BIOS 112, step 220, control is passed to expansion card ROM 112, step 270. However, prior to passing control to expansion ROM 112, a cryptographic hash is computed, step 272, and verified against a stored digital signature for the expansion card, step 25 274. If the signature is valid, then control is passed to the expansion ROM 112 and it is executed.

Id. at 17 (emphases added). The italic portion discloses computing a cryptographic hash for the expansion ROM, and the underling portion discloses verifying, *i.e.* comparing, the cryptographic hash against a digital signature for the expansion ROM. The allocation of the expansion card ROMs and the BIOS are shown in Figure 1a in the Arbaugh Provisional (right) (*Id*. at 20).



221. I note that in the passage I quoted above, the Arbaugh Provisional refers to "expansion ROM 112" instead of "expansion ROM 122" — a POSITA at the time would have understood this is a simple typographical error, as elsewhere in the specification the Arbaugh Provisional refers to "expansion ROM 122. The remainder of the patent is unambiguous that the expansion ROM is 122 and the system BIOS is 112.

222. A POSITA would have also understood that the digital signature in the Arbaugh Provisional is stored in a trusted memory location. The Arbaugh Provisional specifies that the AEGIS BIOS has a trusted portion that is "assumed to be valid" where digital signatures are stored:

> The lowest layer, first layer 200 contains the small section of trusted software, digital signatures, public key certificates, and recovery code AEGIS relies on throughout the boot process. The integrity of layer 200 is assumed to be val*id.*

Id. at 16 (emphasis original). A POSITA would have therefore understood that the digital signature is stored in first layer 200 because it is a "digital signature" which "AEGIS relies on throughout the boot process." Id.

7. <u>Claim 7 of the Arbaugh Patent is Supported by the Provisional</u>

223. Claim 7 of the Arbaugh Patent discloses:

> 7. The method of claim 4, wherein

> step (4) employs a secure protocol to obtain a replacement boot component from a trusted repository to replace said failed boot component.

224. The Arbaugh provisional discloses the further limitation of dependent claim 7 of "step (4) employs a secure protocol to obtain a replacement boot component from a trusted repository to replace said failed boot component." As I discussed above for claim 4, the step of recovering a failed boot component is disclosed by the Arbaugh Provisional's use of a trusted repository:

> Any integrity failures identified in the above process are recovered through a trusted repository, step 298, as discussed below.

Id. at 18. The use of the trusted repository is described in detail as one in which a replacement is obtained for a failed boot component:

> The use of AEGIS ROM 256 as the trusted repository is accomplished through the addition of AEGIS ROM 256, which is an inexpensive PROM board, and modifications to BIOS 112. BIOS 112 and AEGIS ROM 256 contain the verification code, and public key certificates. AEGIS ROM 256 also contains code that allows the secure recovery of any integrity failures found during the initial bootstrap. In essence, the trusted software serves as the root of an authentication chain that extends to the operating system and potentially beyond to application software. If the component that fails its integrity check is a portion of BIOS 112, then it must be recovered from AEGIS ROM 256. *The recovery process is a simple memory copy from the address space of AEGIS ROM 256 to the memory address of the failed component, in effect shadowing the failed component*. A failure beyond BIOS 112 causes the system to boot into a recovery kernel contained on AEGIS ROM 256. The recovery kernel contacts a "trusted" host through a secure protocol, as discussed below, to recover a verified copy of the failed component. *The failed component is then shadowed or repaired, if possible, and the system is restarted.*

Id. at 19-20. A POSITA would have understood that the description of "shadowing" refers to the process of replacing a failed component, because it is described as a "copy from the address space of AEGIS ROM 256 to the memory address of the failed component," which is a replacement of the failed component with a known-good version.

C. Even Assuming the ’941 Patent Were Entitled to a Priority Date Earlier Than October 2, 1997, Arbaugh Conceived of His Patent Before March 1997

225. I have reviewed the Expert Report of Dr. Scott Nettles (including Section C, p. 55-64), as well as the Declaration and Expert Reports provided by Dr. Arbaugh and Dr. Hicks. Based on this information, I agree with Dr. Nettles’s conclusion, and it is also my opinion, that Arbaugh both conceived of his invention before March 1997, and diligently reduced the invention to practice.

1. SwitchWare Project at UPenn

226. As set forth in Dr. Nettles’s report, Dr. Nettles, was a founding member of the DARPA funded SwitchWare project, which was one of the pioneering groups in the area of Active Networking (“AN”). His group developed PLAN, the first domain-specific programming language for programmable packets, as well as PLANet, the first purely active inter-network. Bill Arbaugh’s research was also part of SwitchWare, focused on proving the low-level security infrastructure needed to support Active Networking.

2. The Arbaugh Technical Note Shows Arbaugh Conceived of his Invention at Least as Early as December 1996

227. I have reviewed Dr. Nettles’s analysis of the technical report “A Secure and Reliable Bootstrap Architecture,” published as a technical report at the University of Pennsylvania Department of Computer and Information Science as Technical Report No. MS-CIS-96-35. (“Arbaugh 1996”). I have also reviewed Arbaugh 1996 in connection with this report. I agree with Dr. Nettles’s conclusion, and it is my opinion, that Arbaugh 1996 contains a disclosure showing that by its publication date, Arbaugh and his co-inventors had conceived of the invention disclosed in the Arbaugh Patent.

228. I also agree with Dr. Nettles’s conclusion that Arbaugh 1996 discloses the secure bootstrap architecture including (1) the integrity of components is verified before loading, only

components that have been verified will be executed, and (2) any components that have not been verified can be replaced:

> We define a *guaranteed secure* boot process in two parts. The first is that no code is executed unless it is either explicitly trusted or its integrity is verified prior to its use. The second is that when an integrity failure is detected a process can recover a suitable verified replacement module.

Arbaugh 1996 at 2.

229. The tiered verification process described in the Arbaugh Patent is also described in Arbaugh 1996. This is the process by which a cryptographic hash is calculated for each segment of the program to be executed and compared against a stored signature for that section. Execution only passes when this verification occurs successfully:

> Following successful completion of the checksum, the cryptographic hash of the second section is computed and verified against a stored signature. If the signature is valid, control is passed to the second section, *i.e.*, Level 1.
>
> The second section proceeds normally with one change. Prior to executing an expansion ROM, a cryptographic hash is computed and verified against a stored digital signature for the expansion code. If the signature is valid, then control is passed to the expansion ROM.

Id. at 4.

230. Arbaugh 1996 also discloses the presence of verification codes and key certificates in an erasable portion of non-volatile memory, which are relevant to my invalidity opinion with respect to several limitations:

> The BIOS and the PROM board contain the verification code, and public key certificates. The PROM board also contains code that allows the secure recovery of any integrity failures found during the initial bootstrap.

Id. at 2. Arbaugh 1996 discloses the conditional execution of components that is relevant to my invalidity opinion with respect to several limitations, as it indicates the system can be configured to either continue execution, not execute, or recover failed components before continuing execution:

> When a system detects an integrity failure, one of three possible courses of action can be taken.
>
> The first is to continue normally, but issue a warning. Unfortunately, this may result in the execution or use of either a corrupt or malicious component.
>
> The second is to not use or execute the component. This approach is typically called fail secure, and creates a potential denial of service attack.
>
> The final approach is to recover and correct the inconsistency from a trusted source before the use or execution of the component.

Id. at 2.

231. This entire process, including the use of a network host to recover any components that do not pass integrity verification, is also described by reference to Figure 3 of Arbaugh 1996 (below left), which appears in almost identical form as Figure 2a in the Arbaugh Patent (right).

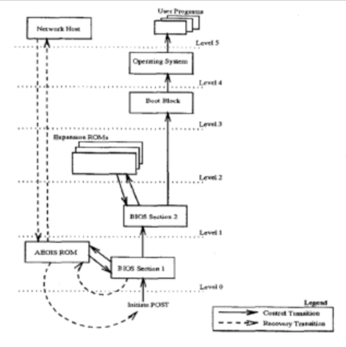


Figure 3. AEGIS boot control flow

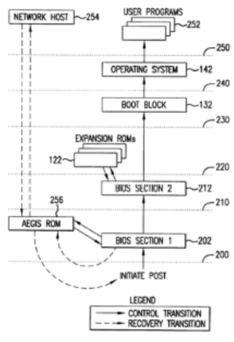


FIG. 2a

232. As noted above, the verification codes and public key certificates are stored in the BIOS memory, which discloses the relevant aspects of the Arbaugh Patent related to storing information used for verifying licenses in the '941 Patent. This is further disclosed by the use of a network host to recover components that are to be replaced:

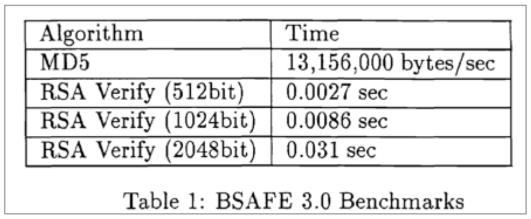
> A failure beyond the BIOS causes the system to boot into a recovery kernel contained on the ROM card. The recovery kernel contacts a "trusted" host through a secure protocol, *e.g.*, IPv6 [1], to recover a verified copy of the failed component. The failed component is then shadowed or repaired, if possible, and the system is restarted.

Id. at 3. In order for failed BIOS components to be restored, those components must reside in an erasable part of the ROM BIOS, as is disclosed in the Arbaugh Patent.

233. The Arbaugh 1996 Article also indicates that verification is achieved through a combination of cryptographic hashes and digital signatures.

> The cost of verification includes time required for computing a MD5 message digest, and the time required to verify the digest against a stored signature.

Id. at 5. A POSITA would have understood the reference to MD5 to be a reference to the well-known message digest algorithm, and the further verification against a stored signature to be a reference to the use of a digital signing algorithm. In the Arbaugh 1996 paper, this is by reference to the time required to complete an RSA Verify (right) (Arbaugh 1996 at Table 1). A POSITA would have recognized this to be the use of then-known RSA public-key cryptography techniques to digitally sign the message digests.



Algorithm	Time
MD5	13,156,000 bytes/sec
RSA Verify (512bit)	0.0027 sec
RSA Verify (1024bit)	0.0086 sec
RSA Verify (2048bit)	0.031 sec

Table 1: BSAFE 3.0 Benchmarks

234. The Arbaugh 1996 Article therefore shows that at that time Arbaugh had conceived of the aspects of the Arbaugh Patent identified below regarding invalidity.

235. I also note Dr. Nettles's statement that the expert reports and declarations of Dr. Arbaugh and Dr. Hicks describe how the Department of Computer Science made technical reports authored by students and faculty available via ftp sites. I concur with his understanding that the Department published the technical reports is consistent with the descriptions provided by Dr. Arbaugh and Dr. Hicks, as that is consistent with how reports authored at the time were submitted and/or transmitted.

3. The Arbaugh April 1997 Article shows Arbaugh Was Reducing his Invention to Practice as of April 1997

236. I have reviewed Dr. Nettles's analysis of the article "A Secure and Reliable Bootstrap Architecture," published in the Proceedings of the 1997 IEEE Symposium on Security

and Privacy ("Arbaugh IEEE 1997"). I have also reviewed Arbaugh 1997 in connection with this report. I understand that this article was presented at an IEEE conference in May 1997, and was authored by William Arbaugh, David Farber, and Jonathan Smith. These are three of the four named inventors on the Arbaugh Patent, which was filed on October 2, 1998 (and claims priority to the Arbaugh Provisional, filed October 2, 1997). I agree with Dr. Nettles's conclusion, and it is my opinion, that Arbaugh IEEE 1997 contains a disclosure showing that, by its publication date, Arbaugh and his co-inventors had conceived of the invention disclosed in the Arbaugh Patent described below regarding invalidity of the '941 Patent.

237. I also understand that Dr. Nettles reviewed a declaration of Dr. Silvia Hall-Ellis regarding the publication date of Arbaugh IEEE 1997. I understand that he concluded that the article was published by IEEE around June-July 1997 time frame after it was presented at the IEEE conference, and I have no reason to doubt that conclusion. Further, this process is consistent with my understanding of the process and timing of how such articles were published at the time.

238. Arbaugh IEEE 1997 in many ways restates and reinforces the work done in the Arbaugh 1996 publication, showing that Arbaugh and his team were continuing to develop an approach to secure digital boot system. Key expansions on the Arbaugh 1996 publication are the further specification of the digital cryptographic techniques being used in the AEGIS system. For example, the RSA Encryption Standard described a method for using public-key cryptography to digitally sign messages. The process for digital signatures described in the RSA Encryption Standard 1993 is the same as in the Arbaugh Patent, in that it generates a message digest (a kind of cryptographic hash), and then encrypts the message digest. Similarly, the Arbaugh Patent discloses using the Digital Signature Standard, DSS, which also uses public-

private key cryptography to encrypt a message digest with a private key in a public-private key pair.

239. Arbaugh IEEE 1997 also discloses the presence of verification codes and key certificates in an erasable portion of non-volatile memory:

> The initial prototype stores the signed cryptographic hashes in a raw format and the public keys in PKCS #1 [13] format. . . .
>
> The last two kilobytes of the 128kb AEGIS BIOS flash ROM contain the component signatures and public key(s).

Id. at 68.

240. The Arbaugh IEEE 1997 Article therefore would have shown to a POSITA that by the time of publication of the Arbaugh 1997 Article that the invention was being diligently reduced to practice.

4. Arbaugh 1997 Technical Report shows Arbaugh Was Reducing his Invention to Practice as of August 1997

241. I have reviewed Dr. Nettles's analysis of the technical report "Automated Recovery in a Secure Bootstrap Process," published as a technical report at the University of Pennsylvania Department of Computer and Information Science as Technical Report No. MS-CIS-97-13. ("Arbaugh Tech. Report 1997"). I understand that this article was published in August 1997, and was authored by William Arbaugh, Angelos Keromytis, David Farber, and Jonathan Smith. These are the four named inventors on the Arbaugh Patent, which was filed on October 2, 1998 (and claims priority to the Arbaugh Provisional, filed October 2, 1997). I agree with Dr. Nettles's conclusion, and it is my opinion, that Arbaugh Tech. Report 1997 contains a description of the invention by Arbaugh and his co-inventors that eventually formed the basis for the Arbaugh Patent.

242. The Arbaugh Tech. Report 1997 is a continuation of the work Arbaugh and three

of co-inventors summarized in the Arbaugh 1996 report. The earlier report disclosed a secure boot process and also referenced a network recovery protocol. The Arbaugh Tech. Report 1997 provides details on how such a protocol would be implemented and how it would be integrated with the secure boot process. The contents of the Arbaugh Tech. Report 1997 and the details of the network recovery protocol were eventually incorporated in the Provisional Application that matured into the Arbaugh Patent.

5. <u>Expert Reports and Declarations of Dr. Arbaugh and Dr. Hicks</u>

243. I have reviewed Dr. Nettles's Expert Report where he notes that he has reviewed the Declaration of Dr. Arbaugh and the Declaration of Dr. Hicks, in addition to the Technical Reports, IEEE publications, the Arbaugh Patent and its file history mentioned above. Similarly, I have reviewed these materials.

244. Based on Dr. Nettles's stated knowledge and work on the SwitchWare project at UPenn in mid- to late-1990s, his familiarity with the research conducted by Dr. Arbaugh and Dr. Hicks, the facts as detailed in the Declaration of Dr. Arbaugh, and the Declaration of Dr. Hicks, and the legal principles provided to me by counsel, I concur with Dr. Nettles's conclusions and it is also my opinion that Dr. Arbaugh conceived of his invention disclosed in his patent at least by December 1996 and worked diligently to file his patent application in fall 1997.

XIII. THE ARBAUGH PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE '941 PATENT

245. I have reviewed Dr. Nettles's analysis of the Arbuagh materials and agree with his conclusion that Arbaugh invalidates the Asserted Claims of the '941 Patent, either literally or inherently, or is obvious in view of the Arbaugh Patent.

246. As mentioned above, the Arbaugh Patent qualifies as prior art under 35 U.S.C. § 102(a) because the Ancora patent is not entitled to any priority earlier than the Israeli patent to

which it claims priority. Even if the Ancora patent were entitled to an earlier conception and reduction, which it does not for the reasons I identified above, the Arbaugh Patent would still qualify as prior art under 35 U.S.C. § 102(g)(2).

A. Claim 1 is Anticipated or Rendered Obvious by the Arbaugh Patent

1. __Limitation 1[preamble] is Disclosed by the Arbaugh Patent__

247. Limitation 1[preamble] of the ’941 patent recites:

> 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:

248. I understand that the Court determined that the portion of the preamble reciting “a method of restricting software operation within a license” is non-limiting, and therefore does not need to be considered for purposes of invalidity. The Court determined that the portion of the preamble reciting “a computer including an erasable, non-volatile memory area of a BIOS of the computer, and a volatile memory area” is limiting.

249. I further understand that the Court construed the term “BIOS” to have its plain and ordinary meaning, wherein BIOS is “an acronym for Basic Input / Output System. It is the set of essential startup operations that begin to run automatically when a computer is turned on, which test hardware, starts the operating system, and support the transfer of data among hardware devices.”

250. I also understand that the Court construed the term “non-volatile memory” to mean “memory whose data is maintained when the power is removed or voltage is too low,” and has construed the term “volatile memory” to mean have its plain and ordinary meaning, which is

"memory whose data is not maintained when the power is removed."[3]

251. The Arbaugh Patent discloses each aspect of the preamble that the Court found limiting: an "erasable, non-volatile memory area of a BIOS of a computer" and a "volatile memory area." Each of these elements is shown in the structural diagram of a typical PC architecture that the Arbaugh Patent describes as the target of the AEGIS process (right) (Arbaugh Patent, Fig. 1c (emphasis added)); see also 5:40-41 ("structural diagram of a typical IBM PC architecture."); 6:39-48 ("AEGIS has been targeted for commercial operating systems on commodity hardware, making it a practical 'real-world' system. . . In the embodiment discussed below, the IBM PC architecture is selected as the platform because of its large user community and the availability of the source code for several operating Systems. FIG. 1(c) is a structural diagram of the typical IBM PC architecture."); Arbaugh Provisional, 10:25-11:2.

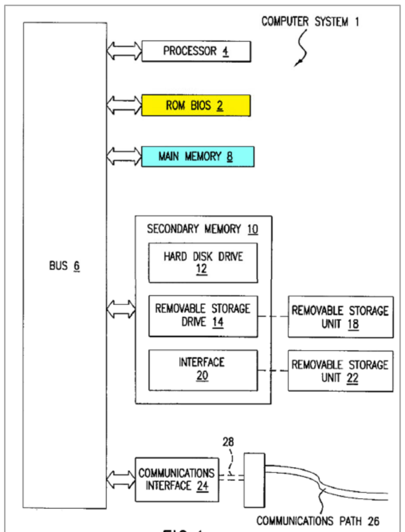


FIG.1c

252. Figure 1(c) of the Arbaugh Patent shows a ROM BIOS 2, highlighted in yellow above. The BIOS memory depicted in Figure 1(c) is a ROM, or read-only memory. The Arbaugh Patent discloses that the BIOS memory is not necessarily a read-only memory, and so can encompass an "erasable, non-volatile memory." For example, claim 1 of the Arbaugh Patent recites "a memory. . . said memory storing a system BIOS for execution by said processor upon

[3] I also understand that as part of this construction the Court determined that "For the corner case where the hard disk drive is used as virtual RAM, the data is not accessible by normal means after the power is removed," but that this clarification is not for presentation to the jury.

power up of the computer system." Arbaugh Patent at claim 1, 22:1-3; Arbaugh Provisional, 11:11-16, 14:8-12, 13:8-17, Fig. 2a. This is "non-volatile" because the contents of the memory, here the system BIOS, is "for execution by said processor upon power up of the computer system." In order for the contents to be executable "upon power up," the contents must already be present. That is, the data in the memory is maintained even when the memory is not being powered.

253. The BIOS memory in the Arbaugh Patent is also erasable. The Arbaugh Patent acknowledges that BIOS ROM can be erased, and a POSITA would have understood that at the time BIOS ROM was regularly updatable by the administrator of a computer. The BIOS ROM could only be updated if it was an erasable ROM, most often as an electrically-erasable read only memory or EEPROM, or flash ROM. The Arbaugh Patent specifically discusses this possibility when it notes that the system ROM BIOS can

> us[e] a flash ROM such as the Intel 28F001BX-B which has an 8KB block that can be protected from reprogramming while the remainder of the ROM can be reprogrammed. Placing the bare essentials needed for integrity verification and recovery in this 8KB block provides a significant level of protection.

Arbaugh Patent, 7:33-38; Arbaugh Provisional, 14:8-18 (disclosing dividing BIOS into two logical sections and verification of second BIOS section), 16:30-17:8 (disclosing replacement of a component failing verification by a valid version), 11:11-16 (disclosing use of PROMs for memory components).[4] The Arbaugh Patent also contemplates that the portion of the ROM

[4] The Arbaugh Provisional does not specifically disclose use of the Intel 28F001BX-B chip to implement BIOS. However, the Arbaugh Provisional discloses that BIOS is split into two "logical" sections, with the first section being the immutable first section of BIOS, and the second section being the replaceable second section of BIOS. Provisional Application, 14:8-18, 15:20-26. A person of ordinary skill in the art at that time knew of flash memory chips that implemented both sections on a single chip. For example, the Intel 28 F0001BX-B chip was marketed and known to people of skill in the art as of 1995. See Intel 1-MBIT (128K x 8) BOOT BLOCK FLASH MEMORY, 28F001BX- T/28F001BX-B/28F001BN-T/28F001BN-B (1995); DEFS_ANCORA00001136- DEFS_ANCORA00001169.

BIOS used for integrity verification and for retrieving components that need to be replaced from a trusted network host can be stored on a programmable read-only memory, or PROM. Arbaugh Patent, 7:38-42, 10:48-56; Arbaugh Provisional, 11:13-16, 16:23-30. A POSITA would have recognized that the EEPROM, flash ROM, or PROM disclosed by the Arbaugh Patent are all erasable, non-volatile memories, and that they are used to store the system BIOS.

254. The Arbaugh Patent also discloses a volatile memory. Main memory 8 shown in Figure 1(c), above, is described as preferably being a "random access memory (RAM)." Arbaugh Patent, 6:50-51; Arbaugh Provisional, 10:25-11:2 (disclosing ordinary PC architecture for the computer, which makes apparent to one of ordinary skill in the art the presence of volatile memory in the form of RAM). A POSITA would have understood that a random access memory being used as a main memory in a computer system is a form of volatile memory.

255. I understand that Ancora has made the argument in its POPR that the scope of the term "memory of the BIOS," which the Court construed to have its plain and ordinary meaning, nonetheless must be further limited, Ancora argued that:

> Absent a showing that the BIOS module actually uses another portion of such physical memory space as part of its normal operations, only the BIOS module would be understood by a person of ordinary skill to be "memory of the BIOS."

Id., at 30. Regardless of whether that interpretation is correct, the Arbaugh Patent discloses a memory space that is used by the BIOS module as part of its normal operations. In particular, the Arbaugh Patent discloses that the BIOS code that checks whether any components need to be replaced, step 261 of Figure 2c, is invoked during the boot process. The operation of this code accesses certificates and public keys that are stored in the BIOS memory identified above, which satisfies the meaning of "memory of the BIOS" even under Ancora's interpretation of the plain and ordinary meaning:

> The lowest layer, first layer 200 contains the small section of trusted software,

> digital signatures, public key certificates, and recovery code AEGIS relies on throughout the boot process. The integrity of layer 200 is assumed to be val*id.* However, after initiating POST, step 260, an initial checksum test is performed, step 262, to identify PROM failures. Second layer 210 contains the remainder of the usual system BIOS code, and the CMOS memory.

Arbaugh Patent, 8:45-53; Arbaugh Provisional, 13:13-18.

2. Limitation 1[a] is Disclosed or Rendered Obvious by the Arbaugh Patent

256. Limitation 1[a] of the '941 patent recites:

> selecting a program residing in the volatile memory

257. I understand that the Court construed the term "program" to mean "a set of instructions that can be executed by a computer," and that the remainder of this limitation has its plain and ordinary meaning. The Court also construed that Limitation 1[a] can be performed at any stage. See Section V, above.

258. The Arbaugh Patent discloses the claimed "program residing in the volatile memory" in the form of bootstrap components that are executed as part of the boot process in the AEGIS model. Arbaugh Patent, 4:40-45; Arbaugh Provisional, 7:7-9. The integrity of these bootstrap components is verified when the system is booted. *Id.* The bootstrap components can also be updated automatically. Arbaugh Patent, 5:3-12; Arbaugh Provisional, 7:29-8:7. When a computer system according to Arbaugh's AEGIS model is booted, there is an initial power-on self-test followed by the execution of a boot-up process that is "under the control of the BIOS 112." Arbaugh Patent, 8:10-11; Arbaugh Provisional, 12:17-19. Arbaugh explains that the code used for the boot process is stored in the two layers BIOS:

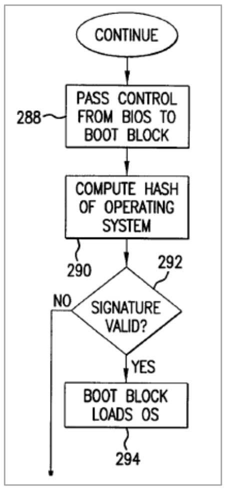
> The lowest layer [of the BIOS], first layer 200 contains the small section of trusted software, digital signatures, public key certificates, and recovery code AEGIS relies on through out the boot process. The integrity of layer 200 is assumed to be valid. However, after initiating POST, step 260, an initial checksum test is performed, step 262, to identify PROM failures. Second layer 210 contains the remainder of the usual system BIOS code, and the CMOS memory.

Arbaugh Patent, 8:45-52 (emphasis added); Arbaugh Provisional, 13:12-23.

259. The boot loading, trusted software, and recovery components of the Arbaugh Patent stored in the BIOS are a "program" because they are executable code, *i.e.* instructions, being invoked by the computer system in order to cause the computer system to perform some function. This therefore meets the Court's construction of "program" to be a set of instructions that can be executed by a computer.

260. The bootstrap components of the Arbaugh Patent also eventually reside in the volatile memory. The Arbaugh Patent discloses that during the power-on self-test (POST) process, the computer executes the code for ordinary boot-up, as well as the customized AEGIS code, both of which are stored in the BIOS. Arbaugh Patent, 8:10-36; Arbaugh Provisional, 12:18-13:6. A POSITA would have understood that in order to execute each one of these BIOS programs, the instructions would be loaded into a volatile memory for execution. Further, after the POST, the BIOS searches for a bootable disk and, when found, loads the code from the bootable disk into memory. Arbaugh Patent, 8:23-31, Arbaugh Provisional, 12:26-13:2.

261. As another example where the Arbaugh Patent loads a program into memory as required by this limitation, one of the final steps of the flow chart shown in Figure 2c is where the AEGIS model loads the operating system into memory (right) (Arbaugh Patent, Figure 2c (excerpt); Arbaugh Provisional Figure 2c. This excerpt shows the step of loading the operating system after the integrity of the operating system has been verified. The step of loading the operating system would have been understood by a POSITA to include loading the operating system program into volatile memory. Even if the operating system were not loaded into volatile memory until after all


CONTINUE
288
PASS CONTROL FROM BIOS TO BOOT BLOCK
COMPUTE HASH OF OPERATING SYSTEM
290
292
NO
SIGNATURE VALID?
YES
BOOT BLOCK LOADS OS
294

verification were performed, it would not matter for purposes of this claim limitation because the court has construed the patent such that the selecting of a program in the volatile memory can occur after the other steps of the method claim happen.

262. In the Final Infringement Contentions, Ancora alleges that the accused LGE mobile product "selects the OTA Update and loads it into RAM." Ancora Final Infringement Contentions, Ex. B, p. 20. Ancora further alleges that the accused LGE mobile products "[are] configured by LGE to perform a method that includes selecting a program residing in such RAM" and that, "following its receipt of an OTA update from an Accused OTA Product, the Accused Android Product selects the OTA update and loads it into RAM." *Id.*, Ex. B, p. 22. Ancora also alleges that the LGE products "write the downloaded update package to an erasable, non-volatile memory area of its BIOS such as /cache/fota/dlpkgfile" *Id.*, Ex. B, p. 27. Ancora further alleges that on LGE non-A/B devices "the downloaded update packages are stored in an erasable, non-volatile memory of the BIOS such as /cache/fota/dlpkgfile and then applied to the device by a program such as /sbin/lge_fota." *Id.* Ancora makes similar allegations regarding LGE's TVs. *See* Ancora Final Infringement Contentions, Ex. D, pp. 13-21. To the extent Limitation 1[a] is broad enough to cover the operation of LGE products accused of infringement as Ancora seems to contend, the specification of the Arbaugh Patent discloses that limitation.

263. This limitation is explicitly disclosed by the Arbaugh Patent, but even if it were not, this limitation would have been rendered obvious to a POSITA in view of the Arbaugh Patent. The Arbaugh Patent discloses the execution of instructions that are stored in the BIOS 112 as part of the bootstrap process, and also discloses executing instructions that are stored on a disk drive

> Once a bootable disk is found, step 164, the bootstrap code loads primary boot sector 132 into memory, step 166, and passes control to it, step 168. The code

> contained in boot sector 132 proceeds to load operating system 142, step 170, or a secondary boot sector (not shown).

Arbaugh Patent, 8:27-31; Arbaugh Provisional, 12:27-13:2. It would have been obvious to a POSITA that the program code in the BIOS and operating system would be loaded into volatile memory prior to selection, for the reason that the cryptographic verification operations that would be run on the program code would be more efficiently and easily performed on code in volatile memory than in either ROM or erasable non-volatile memory. It was conventional at the time in the computer science arts to use volatile memory, such as the main memory of a computer, as a working space when operating on data stored in read-only memories or slower random-access memories such as hard disks or diskette drives in order to improve performance.

3. ***Limitation 1[b] is Disclosed or Rendered Obvious by the Arbaugh Patent***

264. Patent Limitation 1[b] of the ’941 patent recites:

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

265. The Arbaugh Patent discloses an agent that meets this limitation in the form of the software used to implement the AEGIS model. The AEGIS model verifies the integrity of software by computing a cryptographic hash of software to be verified, and then comparing that cryptographic hash to a stored value. Arbaugh Patent, 9:33-39; Arbaugh Provisional, 14:13-18. The cryptographic hash, and associated information, therefore constitutes at least one license record that is part of a verification structure.

266. I understand that the Court construed the term “using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS” to have its plain and ordinary meaning, and that the plain and ordinary meaning of the term “agent” is “a software program or routine.” I further understand that the Court did not limit such a “software program

or routines" to any particular type of software. Thus, because this term is not limited to any particular type of software program or routine, I understand that this term may include both operating system and non-operating system "software programs or routines."

267. I understand that the Court construed the term "set up a verification structure" to mean "establishing or certifying the existence of a pseudo-unique key and establishing at least one license-record location."[5]

268. I understand that Ancora asserts that the limitation of a "pseudo-unique key" can be met by a public key used to decrypt data encrypted by another device. See Ancora Final Infringement Contentions, Ex. B, at 105 ("For example, LGE includes a public key in the hardware and/or non-volatile cache memory of each Accused Android Product . . ."). If Ancora's assertion is correct, and the term "pseudo-unique key" is broad enough so that such a public key can meet the limitation of a pseudo-unique key in the '941 Patent, then the Arbaugh Patent discloses this limitation by virtue of a public key that serves the same purpose and performs the same function as that Ancora relies on in its infringement contentions. *See, e.g.*, Arbaugh Patent, 4:38-45, 9:33-10:3; Arbaugh Provisional, 7:5-11, 14:13-15:12.

269. The cryptographic signature in the Arbaugh Patent discloses the claimed "license record." I understand that the Court construed the term "license record" to mean "data associated with a licensed program with information for verifying that licensed program." Taking the "licensed program" to be the software to be verified, the cryptographic signature in the Arbaugh

[5] I also understand that as part of this construction the Court determined that "Establishing at least one license-record location" may include the steps of "forming a license-record by at least partially encrypting the contents used to form a license-record with other predetermined data contents, using at least part of the pseudo-unique key; and storing the encrypted license-record," but that this clarification is not for presentation to the jury.

Patent is information used to verify the licensed program.[6] As explained in the Arbaugh Patent, when each component—*i.e.* program—is set to be executed, AEGIS first conducts an "integrity check." *E.g.*, Arbaugh Patent, 4:40-43; Arbaugh Provisional, 7:7-9. The integrity check described by the Arbaugh Patent uses a cryptographic hash function to generate information about the program to be executed then compares the resultant hash value with the decrypted digital signature of the program:

> The integrity checks compare a computed cryptographic hash value with a stored digital signature associated with each component.

Arbaugh Patent, 4:43-45, Arbaugh Provisional, 7:10-11.

270. The Arbaugh Patent gives an explicit example of how to prepare a digital signature using the well-known Digital Signature Standard (DSS) and the Secure Hash Algorithm (SHA1). Arbaugh Patent, 15:17-67; Arbaugh Provisional, 25:17-26:25. DSS is an approach to generating a secure signature that can be used to verify that a piece of data (called a "message") has not been modified and authenticates the source. First, a message of arbitrary length is reduced to a single code having a length of 160 bits using the SHA1 algorithm. This code is called a hash value or digest. Arbaugh Patent, 15:51-67; Arbaugh Provisional, 26:14-25.

271. This digest is then "signed." The process of signing uses the special property of public-key cryptography systems that the holder of a (secret) private key can encrypt data that can be decrypted by anyone with the sender's (non-secret) public key. Because only the sender has the private key, and the private key was used to encrypt the message, it follows that only the sender could have encrypted the message. Where the sender is a trusted person or entity, then, this means that a recipient can be reassured that the message was in fact sent by the sender. The

[6] The term "licensed program" has not been construed by the Court and I expressly reserve the right to supplement or amplify my opinion to the extent that any terms in the '941 Patent are subsequently interpreted.

concept of cryptographic signatures is that the message itself does not need to be secret, but that the recipient must nonetheless be assured that the message originated from a trusted sender and has not been tampered with. Rather than applying encryption to the entire message, the encryption is applied only to the digest of the message. This makes the entire process faster. The recipient, then, uses the known public key of the sender to decrypt the digital signature and retrieve the digest of the message. The recipient can then determine its own digest of the same message. If the two digests match, the recipient can know that the message has not been tampered with.

272. In the Arbaugh Patent, the "message" in SHA1 and DSS is the program whose integrity is to be verified. This verification is done by applying the widely-known SHA1 algorithm to the program to be verified and comparing the resulting digest with the decrypted digest received from the sender. In the Arbaugh Patent the format of this information is called a "component signature certificate," and its structure is described as:

```
((cert (issuer (hash-of-key (hash sha1 approverkey)))
        (subject (hash sh1 hashtytes))
        (not-before 09/01/97-0000)
        (not-after 09/06/97-0000)
(signature (hash sha1 hashbytes)
        (public-key dsa-sha1 approverkey)
(sigbytes)))
```

Arbaugh Patent, 14:33-45; Arbaugh Provisional, 23:29-24:13.

273. This certificate therefore includes the claimed "license record" in the form of the signature for the boot components and/or operating system. Consistent with Ancora's interpretation of the claims, which I do not agree with, the signature constitutes a "license record" because it is information that is used to verify that a certain program is authorized to execute, *i.e.*, is "licensed." The signature is also associated with the program because the

signature must be generated by a trusted authority for each program individually and specifically. The signature for one program cannot be used with another.

274. The Arbaugh Patent discloses setting up the verification structure in the "non-volatile memory of the BIOS" as recited in this claim. The Arbaugh Patent discloses that the public key certificates and digital signatures are stored in the AEGIS BIOS:

> The lowest layer, first layer 200 contains the small section of trusted software, digital signatures, public key certificates, and recovery code AEGIS relies on throughout the boot process.

Arbaugh Patent, 8:45-48; Arbaugh Provisional, 13:13-15. The above steps would be performed by software in the Arbaugh Patent that corresponds to the agent in the '941 Patent (see limitation 1[b], above). The Arbaugh Patent also discloses that this section of the BIOS is stored on a single BIOS chip:

> The first assumption is that the motherboard, processor, and a portion of the system ROM (BIOS) are not compromised, *i.e.* the adversary is unable or unwilling to replace the motherboard or BIOS. This assumption can be reduced by using a flash ROM such as the Intel 28F001BX-B which has an 8KB block that can be protected from reprogramming while the remainder of the ROM can be reprogrammed. Placing the bare essentials needed for integrity verification and recovery in this 8KB block provides a significant level of protection.

Arbaugh Patent at 7:30-38; Arbaugh Provisional, 14:8-18 (disclosing dividing BIOS into two logical sections and verification of second BIOS section), 16:30-17:8 (disclosing replacement of a component failing verification by a valid version), 11:11-16 (disclosing use of PROMs for memory components).[7] The Arbaugh Patent further describes storing the first section in a section of the ROM BIOS that can be "protected from reprogramming," that is, would only be rewritable

[7] Although the Provisional Application does not mention the Intel flash memory chip, chips like that which implemented a protected part and an erasable part were known to those of ordinary skill in the art. In fact the Intel 28FOO1BX-B flash memory chip was marketed since at least 1995. Intel 1-MBIT (128K x 8) BOOT BLOCK FLASH MEMORY, 28F001BX-T/28F001BX-B/28F001BN-T/28F001BN-B (1995); DEFS_ANCORA00001136-DEFS_ANCORA00001169. See footnote 4, above.

under limited circumstances "to prevent tampering" by an attacker:

> First section 202 contains the "trusted software", the bare essentials needed for integrity verification and recovery. second section 212 contains the remainder of the system BIOS 112 and the CMOS memory. First section 202 and second section 212 can be contained within a single flash ROM, Such as the Intel 28FOO1BX-B which has an 8KB block that can be protected from reprogramming while the remainder of the ROM can be reprogrammed. Ideally, first section 202 is stored on this 8 KB flash boot block to prevent tampering.

Arbaugh Patent at 9:13-22; Arbaugh Provisional, 14:8-12. These disclosures confirm that both the public key (*i.e.*, the pseudo-unique key that is used in the verification structure of the '941 Patent) and the certificates (*i.e.* the license-records that are used in the verification structure of the '941 Patent) are both present in the memory of the BIOS module.

275. In particular, because (i) the component certificate is embedded within a component (Arbaugh Patent, 14:43; Arbaugh Provisional, 24:12), (ii) the component certificate and component are written to the AEGIS BIOS (*i.e.*, at least the erasable portion of AEGIS BIOS) when the component fails verification and is replaced from the network or a the AEGIS ROM (Arbaugh Patent, 9:11-39 and 10:47-11:15; Arbaugh Provisional, 14:8-18 and 16:23-17:8), and (iii) the component certificate includes both the digital signature of the component and the public key (Arbaugh Patent, 14:33-45; Arbaugh Provisional 23:29-24:13), both the digital signature and the public key may be stored in erasable non-volatile memory that stores BIOS (*i.e.*, the erasable part of the AEGIS BIOS.)

276. In its Final Infringement Contentions, Ancora apparently alleges that Limitation 1[b] is met because a server "is configured by LGE such that it provides an OTA update containing a verification structure to [an LGE mobile unit], which downloads this package and saves it to the cache or data partition of its BIOS." Ancora Final Infringement Contentions, Ex. B, p. 29. Ancora further alleges that the "OTA update includes at least one license record, including payload information such as the application name or version of the program and/or a

'hash' representation of such information that allows the program to be verified." *Id.*, Ex. B, p. 37. Ancora further alleges this limitation is met when "the downloaded update packages are stored in an eraseable, non-volatile memory of the BIOS such as /cache/fota/dlpkgfile. Storing the update package in this location sets up one or more of the previously identified verification structures for use by the update installer." *Id.*, at p. 41. Further, Ancora alleges that "each Accused Android Products is configured by LG to the LGUpdateService::WritePackage function to write a vbmeta structure, including a verification structure, onto a vbmeta partition in the install slot." *Id.*, Ex. B, p. 46-47. See also, *Id.*, Ex. B, pp. 47-50. In other words, Ancora alleges Limitation 1[b] is met when a mobile unit downloads an OTA update and the OTA update contains a "license record" because it contains information such as the application name or version of the program and/or a 'hash' representation of such information. To the extent the Limitation 1[b] is broad enough to cover the operation of LGE products accused of infringement as Ancora seems to contend, the specification of the Arbaugh Patent discloses that limitation. To the extent that this claim term has the scope that Ancora urges, this limitation is disclosed by either client software or server software in the Arbaugh Patent, the Arbaugh Patent discloses server software that generates a digitally signed cryptographic hash of a program, which digitally signed cryptographic hash is transferred to the client in Arbaugh and stored in the non-volatile memory of the BIOS. As described in the Arbaugh Patent, the server has a private key which is used to generate component certificates that digitally sign cryptographic hashes to verify software components. Arbaugh Patent, 12:42-14:58; Arbaugh Provisional, 20:10-24:23. This is transferred to a client along with the program when the component needs to be updated:

> When the certificate expires, the trusted repository of the present invention is contacted and either a new certificate is obtained, in the case where the component does not need an update, or a new component and certificate are obtained, in the case where a newer version of the component is available.

Arbaugh Patent, 5:7-12; Arbaugh Provisional, 8:2-9. The Arbaugh Patent discloses that the private key is attributed to a server, and the server is also responsible for managing communications and transfers with a client:

> When the server receives the "I am booting message from the client, the Server would check a database containing the configuration of the client. The Server would then compare that configuration with the current configuration desired for the client. If they are different, then the server would instruct the client to download the appropriate changes.

Arbaugh Patent, 5:21-26; Arbaugh Provisional, 8:14-18. To the extent that Ancora is asserting a claim scope that encompasses software operating on a server as an "agent" meeting this limitation, that limitation is disclosed by the Arbaugh Patent disclosure of server software generating a private key, a digitally signed cryptographic hash of a component, and the component code, and transferring them to a client.

277. The Arbaugh Patent also discloses client software that is part of the BIOS contact a trusted server before any other operation occurs in order to determine whether any software updates are available. This is shown as step 261 in Figure 2c (below) (Arbaugh Patent, at Fig. 2c (emphasis added)); Arbaugh Provisional, at Fig. 2c. In this annotated copy of Figure 2c from the Arbaugh Patent (originally annotated by Dr. Nettles), outlined is a yellow box those steps that occur under the control of the BIOS and software programs resident in the BIOS. Also filled in yellow the boxes for box 261, labeled "check update server to check for updates," as well as the two boxes that constitute verifying the cryptographic hashes of programs prior to execution, boxes 264 and 272.

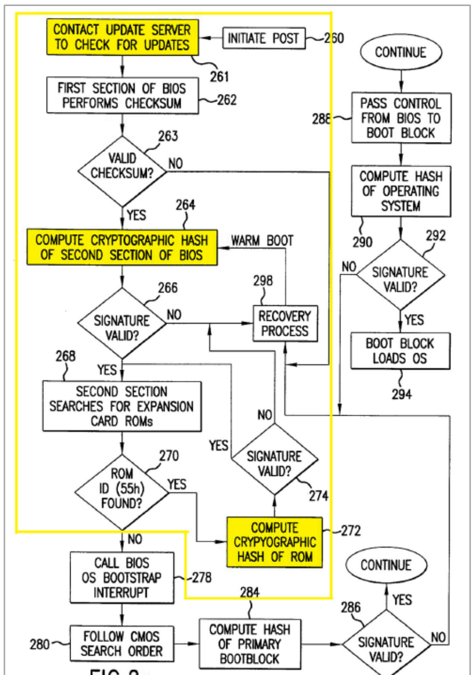

CONTACT UPDATE SERVER TO CHECK FOR UPDATES
INITIATE POST
260
261
CONTINUE
FIRST SECTION OF BIOS PERFORMS CHECKSUM
262
PASS CONTROL FROM BIOS TO BOOT BLOCK
288
263
VALID CHECKSUM?
NO
COMPUTE HASH OF OPERATING SYSTEM
YES
264
COMPUTE CRYPTOGRAPHIC HASH OF SECOND SECTION OF BIOS
WARM BOOT
290
292
SIGNATURE VALID?
298
266
NO
RECOVERY PROCESS
SIGNATURE VALID?
NO
YES
BOOT BLOCK LOADS OS
268
YES
294
SECOND SECTION SEARCHES FOR EXPANSION CARD ROMs
NO
YES
SIGNATURE VALID?
270
ROM ID (55h) FOUND?
YES
274
COMPUTE CRYPYOGRAPHIC HASH OF ROM
272
NO
CONTINUE
CALL BIOS OS BOOTSTRAP INTERRUPT
278
284
YES
286
FOLLOW CMOS SEARCH ORDER
280
COMPUTE HASH OF PRIMARY BOOTBLOCK
SIGNATURE VALID?
NO

FIG.2c

278. Arbaugh describes the process for checking for server updates as follows:

> Step 261 in FIG.2c, to the BIOS to contact the trusted repository after initiating POST, step 260. As is shown in FIG.2c, the remainder of the boot process follows the same procedure as that described for FIG.2b above. The purpose of this contact is two fold. First, it permits a status monitoring of each workstation. Second, the contact allows centralized updates since the trusted repository can respond back with a “I need to update you” message. Step 261 would be done in the following manner. When Server 420 receives the “I am booting message from Client 410, Server 420 would check a database containing the configuration of Client 410. Server 420 would then compare that configuration with the current configuration desired for Client 410. If they are different, then Server 420 would instruct Client 410 to download the appropriate changes.

Arbaugh Patent, at 20:66-21:17; Arbaugh Provisional, 35:25-36:9. According to this disclosure, the BIOS will first initiate the process of checking to see if any software needs to be updated, automatically, as a first step of the boot process after the power-on self test. Thereafter the BIOS will perform integrity checks on each software component, including the second section of the BIOS, prior to execution of those components. This process results in the AEGIS workstation polling the server to determine whether any software needs to be updated; at this point, the code on the workstation is unaware whether any software is out-of-date or needs to be updated, and if the server determines that to be the case, it will direct the trusted software on the workstation to download the appropriate software update.

279. The Arbaugh Patent discloses another mode of operation by which the "agent" can "set up a verification structure" when the agent determines that code on the workstation is corrupted or has expired. The Arbaugh Patent discloses determining whether any data stored for its components, which can include the program itself or its certificates, needs to be replaced:

> The use of network host 254 as the trusted repository is accomplished through the addition of an inexpensive PROM board, and modifications to AEGIS ROM 256. BIOS 112 and AEGIS ROM 256 contain the verification code, and public key certificates. AEGIS ROM 256 also contains code that allows the Secure recovery of any integrity failures found during the initial bootstrap. In essence, the trusted software serves as the root of an authentication chain that extends to the operating system and potentially beyond to application software. If the component that fails its integrity check is a portion of BIOS 112, then it must be recovered from AEGIS ROM 256. The recovery process is a simple memory copy from the address space of AEGIS ROM 256 to the memory address of the failed component, in effect shadowing the failed component. A failure beyond BIOS 112 causes the System to boot into a recovery kernel contained on AEGIS ROM 256. The recovery kernel contacts a "trusted' host through a secure protocol, as discussed below, to recover a verified copy of the failed component. The failed component is then shadowed or repaired, if possible, and the system is restarted.

Arbaugh Patent at 10:47-67; Arbaugh Provisional at 16:23-17:8. This mode is shown in box 298 of Fig. 2c , and is described where the AEGIS BIOS will cause the workstation to request and receive a replacement for software that has not passed its integrity verification:

> Where network host 254 is the trusted repository, the detection of an integrity failure causes the system to boot into a recovery code contained on the AEGIS ROM 256. The recovery code contacts a "trusted' host through the AEGIS recovery protocol, discussed below, to recover a signed copy of the failed component. The failed component is then shadowed or repaired, and the system is

> restarted (warm boot). Note that when the boot process enters the recovery procedure it becomes isomorphic to a secure network boot, except that in AEGIS only the needed bootstrap components are transferred.

Arbaugh Patent, at 11:1-15; Arbaugh Provisional, 17:9-19. In this mode of operation, which is described in great detail in the Arbaugh Patent from columns 11 to 20 (Arbaugh Provisional, 17:9-34:18), the workstation requests a specific program and certificate to be downloaded based on the agent's own determination that the software needs to be updated. The Arbaugh Patent also describes situations where the agent will determine that software has expired because the validity of its authentication certificates has lapsed. In this case, the software will determine that its certificates are no longer valid, and will request replacement certificates in order to continue to authorize the workstation to execute the program:

> Requiring each client to maintain a Certificate Revocation List (CRL) places a significant burden on the non-volatile storage of the client. Rather than use CRLs, the validity period of the certificates can be kept short, as in the SDSL/SPKI model, requiring the client to update the certificates when they expire. This serves two purposes beyond the ability to handle key revocation. First, the storage requirements for CRLs are eliminated. Second, the amount of system maintenance required of the client potentially can be reduced. Since the client must connect to the server on a regular basis to update the component certificates, the server can, at the same time, update the actual component as well if a new version is available.

Arbaugh Patent, at 14:46-58; Arbaugh Provisional, 24:14-23.

280. The Arbaugh Patent discloses that the operation of the AEGIS system can occur in any level of operation of the computer, such as during boot-up, or operating system loading, or during operation of the operating system. For example, the Arbaugh Patent discloses performing a cryptographic hash check for software during execution of the operating system:

> An alternate approach to provide a Secure and Reliable Bootstrap is to move the expansion ROM detection and verification routines, steps 268,270,272, and 274 in FIG.2b, into operating system 142. As is shown in FIG. 2d, after boot sector 132 loads operating system 142, step 294, expansion ROMs 122 can be searched for, step 295, detected, step 296, and verified, step 298, using a computed cryptographic hash, step 297, by the operating system driver interface rather than

> the BIOS. The initialization of operating system 142 is then allowed to continue, step 299.

Arbaugh Patent, 21:19-28; Arbaugh Provisional, 36:10-17. This disclosure explains the operation for generalizing integrity checks for any component (here, expansion ROMs) to occur during execution of the operating system initialization. The Arbaugh Patent further discloses that these steps can occur after operating system initialization, to occur when the operating system interfaces with drivers for expansion ROMs:

> An alternate approach to provide a Secure and Reliable Bootstrap is to modify the above embodiments by moving the expansion ROM detection and verification routines into the operating System. The expansion ROMs can then be detected and verified by the operating system driver interface rather than the BIOS.

Arbaugh Patent, at 5:27-32; Arbaugh Provisional, at 8:19-23.

281. This conclusion still holds in view of Ancora's argument regarding the scope of the "memory of the BIOS" in its POPR. There, Ancora argued that:

> Absent a showing that the BIOS module actually uses another portion of such physical memory space as part of its normal operations, only the BIOS module would be understood by a person of ordinary skill to be "memory of the BIOS."

Id., at 30. Regardless of whether that interpretation is correct, the Arbaugh Patent discloses that the portion of memory where the "agent" sets up a "verification structure" is a memory space that is used by the BIOS module as part of its normal operations. In particular, the Arbaugh Patent discloses that the BIOS code that checks whether any components need to be replaced, step 261 of Figure 2c, is invoked during the boot process. Any updates that occur during this process will necessarily replace the certificates stored for updated components, because certificates for old components will not serve to verify the integrity of new components that have different cryptographic hashes. See Arbaugh Patent, 21:1-14.

282. Additionally, any components replaced during the recovery process, step 298 of Figure 2c, will replace the certificates present in the BIOS. *Id.*, at 20:59-63. This replacement

process is invoked by the execution of the BIOS, using code present in the BIOS, and uses a location in memory on the same physical chip as the BIOS to store replacement certificates. A POSITA would have therefore understood that the Arbaugh Patent discloses the condition that Ancora asserted in its POPR must be met for memory locations on the BIOS memory chip to be a "memory of the BIOS."

283. This limitation is explicitly disclosed by the Arbaugh Patent, but even if it were not, this limitation would have been rendered obvious to a POSITA in view of the Arbaugh Patent. The use of an "agent," which can be invoked as part of the BIOS or part of the operating system, would have been obvious to a POSITA at the time in view of the Arbaugh Patent. This includes the types of "agents" (including API calls) described in the DMI specification that were known to a POSITA at the time. *See*, Section IX.C, above. The Arbaugh Patent discloses that the update process can be invoked by the initial check for updated software, box 261 in Figure 2c, or through the recovery process, box 298 in Figure 2c. This functionality is provided in code that is stored in the BIOS, but it would have been obvious to a POSITA that an agent executed as part of the BIOS, or above the BIOS level as part of the operating system, as part of an API, or part of a user application executing at a level above the operating system could benefit from and invoke the routines to validate the integrity of programs being executed.

284. The Arbaugh Patent provides explicit motivation to use the recovery process of the AEGIS model at a general application level, not limited to the boot process:

> The recovery process discussed above is also easily generalized to applications other than the boot process of the present invention, such as standardized desktop management and secure automated recovery of network elements such as routers or "Active Network" elements.

Arbaugh Patent, 20:37-41; Arbaugh Provisional, 35:3-6. See also Arbaugh Patent, Abstract ("The recovery process is easily generalized to applications other than the bootstrap process of the present invention, such as standardized desktop management and secure automated recovery of network elements such as routers or 'Active Network' elements."). Implementing the recovery process described for program code other than at the boot process level would involve nothing more than applying the teachings of the Arbaugh Patent to different layers of code within the same computer. Indeed, the iterative process disclosed for the Arbaugh Patent would make such a modification straightforward for a skilled artisan, and would not require undue experimentation. A POSITA would have been motivated to attempt to use a similar approach to apply integrity checks to general user applications, as suggested by the Arbaugh Patent at 20:37-41, using an approach such as that taught by the Arbaugh Patent at 21:19-28, in order to improve the robustness of the operating system, improve system management, and reduce total cost of ownership, as discussed generally in the Arbaugh Patent at column 21.

4. <u>Limitation 1[c] is Disclosed by the Arbaugh Patent</u>

285. Limitation 1[c] of the '941 patent recites:

> verifying the program using at least the verification structure from the erasable non-volatile memory of the BIOS

286. Arbaugh discloses that AEGIS verifies the integrity of software by computing a cryptographic hash of software to be verified, and then comparing that cryptographic hash to a stored value. Arbaugh Patent, 9:33-39; Arbaugh Provisional, 14:13-18. This uses the verification structure discussed for the preceding limitation.

287. I understand that the Court construed the term "verifying the program using at least the verification structure" to mean "confirming whether a program is licensed using at least the verification structure." In the context of the present litigation, I understand that Ancora contends that a program being "licensed" means that the program is authorized to be executed.

288. As explained by the Arbaugh Patent, AEGIS uses a "chain of integrity checks."

Arbaugh Patent, 4:40-45; Arbaugh Provisional, 7:7-7:11. The Arbaugh Patent provides examples of "integrity" checks consistent with the understanding of a person having ordinary skill in the art, including checking that the software has not been accidentally or maliciously replaced with unauthorized software in order to alter the operation of the computer system:

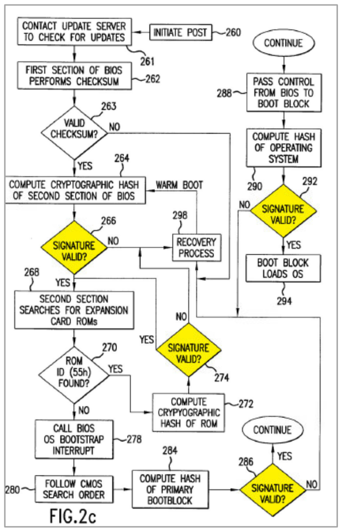
> The AEGIS integrity policy prevents the execution of a component if its integrity cannot be validated. There are three reasons why the integrity of a component could become inval*id.* The integrity of the component could change because of some hardware or software malfunction, the integrity of the component could change because of some malicious act, or the component's certificate time stamp may no longer be val*id.*

Arbaugh Patent, at 10:13-19; see also *id.* at 1:32-41; Arbaugh Provisional, 15:19-25; see also *id.* at 1:21-2:4. It would have been apparent to a POSITA that "integrity" relates to whether software is authorized to execute by reference to the courses of action that are taken when the integrity checks are failed. Specifically, the Arbaugh Patent notes that:

> When a system detects an integrity failure, one of three possible courses of action can be taken. The first is to continue normally, but issue a warning. Unfortunately, this may result in the execution of either a corrupt or malicious component. The second is to not use or execute the component. This approach is typically called fail secure, and creates a potential denial of service attack. The final approach is to recover and correct the inconsistency from a trusted repository before the use or execution of the component.

Arbaugh Patent at 3:26-35; Arbaugh Provisional, 5:1-7. It is immediately apparent that these two courses of action (as well as the third, which is to replace the software with an authorized version and is discussed below) show that code that lacks "integrity" is not permitted to be executed, which is the same unauthorized or unlicensed software: all three are instances of software that as a matter of policy are not permitted to be executed.

289. AEGIS performs this integrity check by calculating the cryptographic hash of each program that is executed and comparing this cryptographic hash against a stored signature. Arbaugh Patent, 9:33-60; Arbaugh Provisional, 14:13-15:5. The Arbaugh Patent shows in Figures 2b and 2c, which perform this verification process for programs in the BIOS, expansion card ROMs, boot blocks, and operating system (right) (Arbaugh Patent, Fig. 2c, emphasis added); Arbaugh Provisional, Fig. 2c. Each of blocks 266, 274, 286, and 292 performs the check of comparing the cryptographic hash calculated from the programs to be loaded (calculated in blocks 264, 272, 284, and 290) against the respective digital signatures stored in memory. Arbaugh Patent, at 9:33-60, 20:66-21:18; Arbaugh Provisional, 14:13-15:5, 35:25-36:9.


CONTACT UPDATE SERVER TO CHECK FOR UPDATES
INITIATE POST
260
261
CONTINUE
FIRST SECTION OF BIOS PERFORMS CHECKSUM
262
288
PASS CONTROL FROM BIOS TO BOOT BLOCK
263
VALID CHECKSUM?
NO
YES
COMPUTE HASH OF OPERATING SYSTEM
264
COMPUTE CRYPTOGRAPHIC HASH OF SECOND SECTION OF BIOS
WARM BOOT
290
292
NO
SIGNATURE VALID?
266
298
YES
SIGNATURE VALID?
NO
RECOVERY PROCESS
BOOT BLOCK LOADS OS
268
YES
294
SECOND SECTION SEARCHES FOR EXPANSION CARD ROMs
NO
270
YES
SIGNATURE VALID?
ROM ID (55h) FOUND?
YES
274
NO
COMPUTE CRYPYOGRAPHIC HASH OF ROM
272
CONTINUE
CALL BIOS OS BOOTSTRAP INTERRUPT
278
284
YES
286
280
FOLLOW CMOS SEARCH ORDER
COMPUTE HASH OF PRIMARY BOOTBLOCK
SIGNATURE VALID?
NO
FIG.2c

290. In its Final Infringement Contentions, Ancora alleges that Limitation 1[c] is met because each LGE mobile unit allegedly "is configured by LGE such that it reboots into recovery mode and then verifies the OTA update." Ancora Final Infringement Contentions, Ex. B, p. 51. Ancora then points to step 5 of the Life of an OTA update that verifies the cryptographic signature of the package. *Id.* Ancora also alleges that Limitation is met because LGE mobile unit allegedly "is configured by LGE such that it reboots and then utilizes the above described VBMeta and dm-verity to verify the boot." *Id.*, Ex. B, p. 56. Ancora then points to step 7 of the Life of an OTA update. *Id.*, Ex. B, pp. 56-57. When describing the operation of LGE's mobile units, Ancora alleges that the LGE mobile units "include an update installer that verifies the program before permitting that program to be installed onto the device. The update installer

verifies whether the program has been signed by an authorized key, indicating that it is licensed to be applied to this device." *Id.*, Ex. B, p. 59. Ancora further alleges that the LGE mobile products are "configured such that, calling through mDmcFotaInerface.Validation and Native.ValidatePackage, control reaches the function FOTAJNI_ValidatePkg. The function verifies that the RSA signature attached to the package corresponds to the contents of the package and that the signature was produced by a trusted entity possessing the private part of a pseudo-unique key." *Id.* Ancora further alleges that the LGE mobile units are "configured by LG to have a bootloader that verifies the program before running it. The bootloader is configured to verify whether the program has been signed by an authorized private key, indicating that it is licensed to operate on this device." *Id.*, Ex. B, p. 60; see also pp. 60-61. To the extent Limitation 1[c] is broad enough to cover the operation of the Accused LGE products as Ancora seems to contend, the specification of the Arbaugh Patent discloses that limitation because Arbaugh uses a digitally-signed cryptographic hash (*i.e.*, what Ancora accused is a "license record" in its infringement contentions) stored with a public key (*i.e.*, what Ancora accuses is a "pseudo-unique key") stored in a memory of the BIOS.

5. Limitation 1[d] is Disclosed or Rendered Obvious by the Arbaugh Patent

291. Limitation 1[d] of the '941 patent recites:

acting on the program according to the verification

292. I understand that the Court construed the term "acting on the program according to the verification" to have its plain and ordinary meaning, and that the plain and ordinary meaning includes that the step of "acting on the program" may include but is not limited to "restricting the program's operation with predetermined limitations, informing the user on the unlicensed status, halting the operation of the program under question, and asking for additional user interactions."

293. The Arbaugh Patent discloses "acting on the program according to the verification" in the form of preventing the execution of programs whose integrity cannot be validated, *i.e.*, verified. Arbaugh Patent, 10:13-14 ("The AEGIS integrity policy prevents the execution of a component if its integrity can not [sic] be validated."); Arbaugh Provisional, 15:20-21. A POSITA would have understood that preventing execution and halting operation mean the same thing in the context of a technology like the '941 Patent. When a program cannot be verified, and no trusted replacement for the non-verified program can be obtained, then the Arbaugh Patent also discloses that operation may continue with predetermined limitations. Arbaugh Patent, 10:23-25 ("For instance, a user may choose to continue operation in a limited manner."); Arbaugh Provisional, 15:28-16:2.

294. In its Final Infringement Contentions, Ancora alleges that Limitation 1[d] is met because an LGE mobile unit is allegedly "configured by LGE such that it will install the update only if its verified." Ancora Final Infringement Contentions, Ex. B, p. 61. Ancora points to steps 5 to 7 of the Life of an OTA update. *Id.* When discussing the operation of LGE mobile units, Ancora alleges that the LGE mobile units "have an update installer that acts on the program according to the verification" and are "configured by LG to include a bootloader that acts on the program according to the verification" *Id.*, Ex. B, p. 68-71. To the extent the Limitation 1[d] is broad enough to cover the operation of LGE products accused of infringement as Ancora seems to contend, the specification of the Arbaugh Patent discloses that limitation.

295. This limitation is explicitly disclosed by the Arbaugh Patent, but even if it were not, this limitation would have been rendered obvious to a POSITA in view of the Arbaugh Patent. The Arbaugh Patent discloses stopping execution or allowing limited execution to continue if an integrity check fails, as noted above. Arbaugh Patent, 10:13-14, 10:23-25,

Arbaugh Provisional, 15:20-21, 15:28-16:2. It would have been obvious to a POSITA that other potential reactions to an integrity verification failure would be possible and desirable, including informing the user on the unlicensed status or asking for additional user interactions. For example, the Arbaugh Patent explicitly discloses halting execution, however, a POSITA would have recognized that it would be customary, when designing a computer program to cease operation, to notify the user why the program has halted. Including a notice to the user prior to halting that indicates that there was an integrity verification failure would be trivial for a skilled artisan to implement in the AEGIS system of the Arbaugh Patent. Providing a user the option to provide additional input would also be obvious to a person having ordinary skill in the art, as the Arbaugh Patent already provides two different choices and a user may have a preference to use one choice over another (allowing limited execution, say, instead of terminating execution). A skilled artisan would recognize that users of computer systems may have different goals or priorities in their use of generalized computer systems and may, at times, prefer to execute a limited set of functionality that will still allow them to accomplish that purpose. As just one example, if a user wished to access a computer system to review a work document, and an unrelated expansion ROM verification failed, the user might prefer to be allowed to review the work document without being able to use the expansion ROM rather than having the entire computer disabled due to the integrity verification failure. Addressing the possibility of multiple preferences would motivate a person having skill in the art to include a request for user input to determine how to proceed, among choices configured by the system administrator, for the reasons identified above.

B. Claim 2 is Anticipated by the Arbaugh Patent

296. Limitation 2 of the ’941 patent recites:

A method according to claim 1, further comprising the steps of: establishing a license authentication bureau.

297. The Arbaugh Patent renders claim 2 invalid because it discloses all of the limitations discussed above for claim 1, from which claim 2 depends, and also discloses the step of “establishing a license authentication bureau” in dependent claim 2.

298. I understand that the parties have not offered a construction of the term “license authentication bureau,” nor has the Court construed the term. The ’941 Patent describes the role of the license authentication bureau broadly, for example that it is a “telecommunications accessible processor where functions such as formatting, encrypting, and verifying may be performed.” ’941 Patent, 3:42-44. The functions can also include that it “might maintain a registry of keys and of licensed programs that have been registered at the bureau in association with these keys.” *Id.*, at 4:19-21. These are all optional, however, as the specification states:

> Those versed in the art will readily appreciate that the invention is by no means bound by the data, the algorithms, or the manner of operation of the bureau. It should be noted that the tasks of establishing and/or verifying a license record may be shared between the bureau and the computer, done exclusively at the computer, or done exclusively at the bureau.

Id., 4:29-35. Moreover, the license authentication bureau may not be a remote system but can also be part of the computer system implementing the invention. *Id.*, at 6:1-3 (“by way of non-limiting example, the bureau, instead of being external entity may form part of the computer.”).

299. Ultimately the only defining characteristic of a license authentication bureau that would have been understood by a POSITA in view of the specification is that, in a preferred embodiment, it:

> can participate in either or both of: (i) establishing the license record in the second non-volatile memory; and
>
> (ii) verifying if the key and license record in the non-volatile memory(s) is compatible with the license record information as extracted from the application under question.

Id., at 3:34-41. The Arbaugh Patent states that the license authentication can participate in either of both of these tasks and, therefore, participation in any way in establishing the license record in the second non-volatile memory or verifying if the license record is compatible with extracted license record information suffices. This is then confirmed by the claims, which make reference to a license authentication bureau without any further restrictions on the properties or operation of that element, such as claim 2.

300. The additional limitation of claim 2 is disclosed by the Arbaugh Patent. In particular, it can be disclosed either by the trusted repository found in network host 254, or in the software elements of the AEGIS ROM 256 itself. Neither of these disclosures strictly depends on the understanding of the term "license authentication bureau" described above. Rather, they both depend on the general contours of how the term "license authentication bureau" would have been understood by a POSITA in view of the specification, with particular emphasis on the specification passages I have referred to earlier.

301. Network host 254 discloses the claimed license authentication bureau because it participates both in establishing a license record (*e.g.* cryptographic hash) in the non-volatile memory, and in verifying if the license record is compatible with the program (*e.g.* confirms that the program is authorized to be executed). The Arbaugh Patent explains that in its embodiment using network host 254, then when a cryptographic hash of a program does not match a program, AEGIS will contact a trusted host to download a verified copy of that program:

> The recovery kernel contacts a "trusted' host through a secure protocol, as discussed below, to recover a verified copy of the failed component. The failed component is then shadowed or repaired, if possible, and the system is restarted.

Arbaugh Patent, 10:63-67; Arbaugh Provisional, 17:5-8.

302. A POSITA would have understood that this constitutes establishing a license record because the license record includes a cryptographic hash of the program, and it would be

evident that when retrieving the "verified copy of the failed component" then the failed component would include both the program instructions and a copy of the cryptographic hash. This is because a POSITA would have recognized that the verification function could have failed for at least two possible reasons: one is that the program instructions could have been modified (so that the cryptographic hash of the program does not match a valid license key), and another is that the verification code has been corrupted or modified, or has expired (so that the cryptographic hash of a valid program does not match the erroneous certificate):

> There are three reasons why the integrity of a component could become inval*id.* The integrity of the component could change because of some hardware or software malfunction, the integrity of the component could change because of some malicious act, or the component's certificate time Stamp may no longer be valid.

Arbaugh Patent, 10:13-19; Arbaugh Provisional, 15:21-25. Therefore, when recovering the failed component, both the program instructions and the verification code will be retrieved anew. When the new verification code is retrieved, it would need to replace the (possibly corrupted) verification code in the non-volatile memory, and in this way participate in establishing a license record.

303. Network host 254 also participates in verifying that the key and license record in the non-volatile memory is compatible with the license record information, when it is used to provide a new program and/or verification code to assist with verification.[8] Providing the program is "participating" in verifying that the application matches the license key because it provides a component that is then compared to the license key in order to confirm that it was correctly received.

[8] As noted earlier, a POSITA would have understood that the network host 254 would be expected to send both the program instructions and certificate when providing a verified copy of a failed component. For purposes of participation in "verification," only delivery of the program instructions is necessary.

304. AEGIS ROM 256 also discloses the claimed license authentication bureau because it participates both in establishing a license record (*e.g.* cryptographic hash) in the eraseable non-volatile memory, and in verifying if the license record is compatible with the program (*e.g.* confirms that the program is authorized to be executed). AEGIS ROM 256 is "established" by someone by virtue of the decision to include it in the AEGIS computer system. The AEGIS ROM 256 includes cryptographic hash in the non-volatile BIOS memory:

> BIOS 112 and AEGIS ROM 256 contain the verification code, and public key certificates. AEGIS ROM 256 also contains code that allows the secure recovery of any integrity failures found during the initial bootstrap.

Arbaugh Patent, 10:49-53; Arbaugh Provisional, 16:25-28. AEGIS ROM 256 participates in establishing the verification codes in several ways. One way AEGIS ROM 256 participates in establishing the verification codes is to store the verification codes, as described above. Another way AEGIS ROM 256 participates in establishing the verification codes is when it is used to store backup copies of verification codes that are restored to other parts of the ROM. For example, earlier I discussed how the trusted repository can be stored on the network in network host 254. The Arbaugh Patent also allows that the trusted repository can be stored in AEGIS ROM 256:

> AEGIS attempts to recover from a trusted repository, Step 298, as discussed below. . . . If the component that fails its integrity check is a portion of BIOS 112, then it must be recovered from AEGIS ROM 256. The recovery process is a simple memory copy from the address space of AEGIS ROM 256 to the memory address of the failed component, in effect shadowing the failed component.

Arbaugh Patent, 10:19-61; Arbaugh Provisional, 15:25-17:4. This passage in the Arbaugh Patent confirms that AEGIS ROM 256 can copy a component, which can include the verification code, into BIOS 112, which is a non-volatile memory that, as noted earlier, also contains verification codes. Arbaugh Patent, 10:49-53; Arbaugh Provisional, 16:26-28.

305. Whether the trusted repository is the AEGIS ROM 256 or is a trusted network

host, in either case the Arbaugh Patent discloses that the trusted repository exists, which a POSITA would have understood means that it must have been "established" at some point by the person implementing the AEGIS system.

306. In its Final Infringement Contentions, Ancora alleges that Claim 2 is met because LGE has allegedly established servers and has allegedly configured the servers to transmit an OTA Update package to a LGE mobile unit. Ancora Final Infringement Contentions, Ex. B, pp. 71-75. To the extent the Claim 2 is broad enough to cover the operation of LGE products accused of infringement as Ancora seems to contend, the specification of the Arbaugh Patent discloses that limitation.

C. Claim 3 is Anticipated by the Arbaugh Patent

1. __Limitation 3[a] is Disclosed by the Arbaugh Patent__

307. Limitation 3[a] of the '941 patent recites:

> A method according to claim 2, wherein setting up a verification structure further comprising the steps of establishing, between the computer and the bureau, a two-way data-communications linkage

308. The Arbaugh Patent discloses one embodiment where the verification process involves establishing a network connection to a trusted network host 254 from the AEGIS ROM 256. *See* Arbaugh Patent, at 10:12-11:24; Arbaugh Provisional, 15:20-17:26.

309. In this embodiment, the Arbaugh Patent describes that the AEGIS ROM 256 is invoked during a failure to validate the integrity of the BIOS 112:

> A failure beyond BIOS 112 causes the System to boot into a recovery kernel contained on AEGIS ROM 256. The recovery kernel contacts a "trusted' host through a Secure protocol, as discussed below, to recover a verified copy of the failed component.

Arbaugh Patent, 10:61-65; Arbaugh Provisional, 17:4-7. This process is a step of "setting up the verification structure" because it is part of the process by which a license-record is set up in the memory of the BIOS, as described above for limitation 1[b].

310. The trusted network host 254 discloses the limitation of "the bureau," *i.e.* the "license bureau" claimed in claim 2, for the reasons I explained for claim 2 above.

311. The communication between the AEGIS ROM 256 and network host 254 discloses a "two-way data-communications linkage" which is a communication channel by which the AEGIS ROM 256 transmits a request for a replacement program and verification code, and by which the network host 254 can provide the requested program and verification code:

> The recovery code contacts a "trusted' host through the AEGIS recovery protocol, discussed below, to recover a signed copy of the failed component.

Arbaugh Patent, 11:3-6; Arbaugh Provisional, 17:11-13. This is explained in more detail in the later discussion of the AEGIS Recovery Protocol, which includes a discussion of the use of the trivial file transfer protocol, TFTP, to request and download a new component:

> Once the DHCP protocol is completed and the shared secret established, the AEGIS client contacts the trusted repository using TFTP with authentication and downloads the new component.

Arbaugh Patent, 19:42-45; Arbaugh Provisional, 33:12-14.

312. A POSITA would have recognized that this disclosure in the Arbaugh Patent describes the computer (*i.e.* the AEGIS ROM in Arbaugh) establishing a two- way data-communications linkage (*i.e.* a network connection in Arbaugh) with a license bureau (*i.e.* a trusted host in Arbaugh). The trusted host in Arbaugh provides a new verification code for a component of BIOS 112 and is therefore part of set up a verification structure in the erasable, non-volatile memory of the BIOS, as required by claim 1.

2. <u>Limitation 3[b] is Disclosed by the Arbaugh Patent</u>

313. Limitation 3[b] of the '941 patent recites:

> 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

314. The Arbaugh Patent discloses this limitation in the same passage describing a network bootstrap method that I referenced in the preceding limitation. Specifically, the Arbaugh Patent describes shadowing a failed BIOS component by obtaining a new copy of the program from a trusted host over a network connection:

> AEGIS ROM 256 also contains code that allows the secure recovery of any integrity failures found during the initial bootstrap. In essence, the trusted software serves as the root of an authentication chain that extends to the operating system and potentially beyond to application software. If the component that fails its integrity check is a portion of BIOS 112, then it must be recovered from AEGIS ROM 256. The recovery process is a simple memory copy from the address space of AEGIS ROM 256 to the memory address of the failed component, in effect shadowing the failed component. A failure beyond BIOS 112 causes the system to boot into a recovery kernel contained on AEGIS ROM 256. The recovery kernel contacts a “trusted’ host through a secure protocol, as discussed below, to recover a verified copy of the failed component.

Arbaugh Patent, 10:51-65; Arbaugh Provisional, 16:26-17:7.

315. The Arbaugh Patent discloses transmitting a request from the AEGIS ROM 256 to a network host 254 in order to obtain a replacement for a component that has failed its integrity verification. The Arbaugh Patent discloses transmitting this request using an authenticated DHCP/TFTP protocol in its recovery protocol:

> Note that when the boot process enters the recovery procedure it becomes isomorphic to a Secure network boot, except that in AEGIS only the needed bootstrap components are transferred. This fact is leveraged by adding authentication to the well-known network protocols Supporting the boot process (DHCP and TFTP) and using them as the recovery protocol . . .

Arbaugh Patent, 11:8-15; Arbaugh Provisional, 17:14-19.

316. This transmission includes an identification of the computer. Arbaugh Patent, 17:32-34, 17:43-46; Arbaugh Provisional, 29:21-23, 29:29-30:1.

317. This transmission also includes an identification of the license-record's contents from the selected program. The Arbaugh Patent discloses that a computer exchanges DHCP

messages with a server in order to authenticate its identity, and in the process transmits the name of a required component as well as the SHA1 hash of the component in the request:

> In AEGIS, the client follows the DHCP protocol, as shown in FIG. 4, but adds to the DHCPDISCOVER message, Step 432, the name of the required component needed followed by the SHA1 hash of the component in the boot file name field, 336. Once the DHCP protocol is completed and the shared secret established, the AEGIS client contacts the trusted repository using TFTP with authentication and downloads the new component.

Arbaugh Patent, 19:38-45; Arbaugh Provisional, 33:8-14. The SHA1 hash of the component sent in the request is the "license-record's contents from the selected program" because the license-record includes a cryptographic hash of the component to be requested in its transmissions to the trusted network host. As described above for claim 1 the license-record's contents include the cryptographic hash, which may be further digitally signed.

3. __Limitation 3[c] is Disclosed by the Arbaugh Patent__

318. Limitation 3[c] of the '941 patent recites:

> 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

319. I understand that Ancora contends that this limitation can be met by having a cryptographic signature applied to a digest of a computer program. Ancora Final Infringement Contentions Ex. B, at 85-93. Based on my review of those contentions, Ancora appears to contend that "forming an encrypted license record at the bureau by encrypting parts of the request-for-license" is met by using information from the program whose integrity is to be verified. Ancora references forming an encrypted license

> by encrypting parts of the request-for-license using part of the identification of the Accused Android Product and the license- record's contents from the selected program, including "build" information (https://developer.andro*id*.com/reference/android/os/Build), as an encryption key.

Id. at 125. I understand from this that in order to meet this limitation, according to Ancora, what must be shown is that an integrity verification code is derived from the program whose integrity is to be verified. I also understand that Ancora contends that "using part of the identification as an encryption key" can be met by using a private key of a trusted authority to sign the relevant integrity certification, and using a public key to verify the signature. *Id.* at 134-142.

320. Based on Ancora's assertions regarding the scope of this limitation, the Arbaugh Patent discloses this limitation. The Arbaugh Patent discloses that replacement components are received either by copying over from a "known-good" copy of the ROM, or from a network host:

> AEGIS ROM 256 also contains code that allows the secure recovery of any integrity failures found during the initial bootstrap. In essence, the trusted software serves as the root of an authentication chain that extends to the operating system and potentially beyond to application software. If the component that fails its integrity check is a portion of BIOS 112, then it must be recovered from AEGIS ROM 256. The recovery process is a simple memory copy from the address space of AEGIS ROM 256 to the memory address of the failed component, in effect shadowing the failed component. A failure beyond BIOS 112 causes the system to boot into a recovery kernel contained on AEGIS ROM 256. The recovery kernel contacts a "trusted" host through a secure protocol . . . to recover a verified copy of the failed component. The failed component is then shadowed or repaired, if possible, and the system is restarted.

Arbaugh Patent, 10:51-67; Arbaugh Provisional, 16:26-17:8. The Arbaugh Patent is explicit that the downloaded component is a signed copy, and that the component is accompanied by a signed certificate to verify the component's integrity:

> When the certificate expires, AEGIS contacts the trusted repository and either obtains a new certificate in the case where the component does not need an update, or a new component and certificate in the case where a newer version of the component is available.

Arbaugh Patent, 20:59-63; Arbaugh Provisional, 35:20-23. The Arbaugh Patent expands on this with its pseudo-code that describes replacing a certificate from a trusted repository Arbaugh Patent, 10:30-42, 14:50-51 ("requiring the client to update the certificates when they expire"); Arbaugh Provisional, 16:5-19, 24:17.

321. A POSITA would have understood that, if one were to assume that the claim has

the scope Ancora seems to assert in its infringement allegations, the modification detection code is "form[ed] . . . at the license bureau" because the Arbaugh Patent discloses that the digitally signed verification certificates are received from the trusted authority:

> When the certificate expires, the trusted repository of the present invention is contacted and either a new certificate is obtained, in the case where the component does not need an update, or a new component and certificate are obtained, in the case where a newer version of the component is available.

Arbaugh Patent, 5:7-12; Arbaugh Provisional, 8:4-7.

322. A POSITA would have understood that, if one were to assume that the claim has the scope Ancora seems to assert in its infringement allegations, the component certificate "encrypt[s] part of the request-for-license" because the request-for-license identifies a program that is to be downloaded.

323. A POSITA would have understood that, if one were to assume that the claim has the scope Ancora seems to assert in its infringement allegations, the modification detection code digitally signed with the private key of the trusted authority "us[es] part of the identification as an encryption key" because under Ancora's theory the combination of the public key and private key serves as an identification, and the private key is therefore "part of the identification." In the Arbaugh Patent a private key is used as an encryption key to digitally sign cryptographic hashes, which suffices to disclose this limitation under Ancora's infringement theory.

324. In particular, in its Final Infringement Contentions, Ancora contends that a FOTA server will encrypt the OTA Update package with a key corresponding to the accused LGE mobile product's model number and/or current Android version. Ancora Final Infringement Contentions, Ex. B, p. 85. Ancora further alleges that a server also signs a LGE-approved executable of a boot component using a private key, potentially using "key0", although Ancora has not yet identified the specific code. Ancora Final Infringement Contentions, Ex. B, p. 121;

130.

325. Based on Ancora's contentions, signing with a private key of a trusted authority meets this limitation, notwithstanding that the private key of a trusted authority is not part of an "identification" of the computer on which the software is being run, as this claim requires. Ancora's contention appears to be based on the assertion that a private/public key pair can be considered two parts of a single encryption key, and that the use of a private/public key pair to sign a given program means that the encryption keys "identif[y]" that program:

> This is demonstrated by the below, which describes how the VBMeta struct is signed by one or more keys corresponding to the identification provided by the Accused Android Product:

Ancora Final Infringement Contentions Ex. B, at 85; 154. If Ancora is correct that the identification of a program is deemed to identify a private/public key pair on a trusted authority, then the Arbaugh Patent discloses this limitation because it discloses using an appropriate private key at the trusted authority to digitally sign the modification detection code for each program that is requested:

> In a capability based model, the certificate itself carries the authorizations of the holder eliminating the need for an identity infrastructure and access control lists. In AEGIS, two capabilities, SERVER and CLIENT, are used with the obvious meanings.
> . . .
> This certificate, signed by a trusted third party or certificate authority, grants to the keyholder (the machine that holds the private key) the capability to generate the second type of certificate, an authentication certificate. The authentication certificate demonstrates that the client or server actually hold the private key corresponding to the public key identified in the authentication certificate.
> . . .
>
> Since the client must connect to the Server on a regular basis to update the component certificates, the Server can, at the same time, update the actual component as well if a new version is available.

Arbaugh Patent at 13:34-38, 13:50-57, 14:55-58.; Arbaugh Provisional at 21:31-35, 22:9-14, 24:20-24. These disclosures, as well as the disclosure of the component certificate at Arbaugh

Patent 14:33-45 and Arbaugh Provisional 23:29-24:11, explain that the certificates issued for components in the AEGIS model are signed by the server using the server's own private key. The server refers to the trusted repository. Arbaugh Patent 17:32-34; Arbaugh Provisional at 29:21-23. The public key used by the computer is used to decrypt components signed with the private key of the server, and therefore the Arbaugh Patent discloses the same configuration that Ancora contends meets this limitation.

4. __Limitation 3[d] is Disclosed by the Arbaugh Patent__

326. Limitation 3[d] of the '941 patent recites:

> transferring, from the bureau to the computer, the encrypted license record

327. In its Final Infringement Contentions, Ancora alleges that the FOTA update is transferred and downloaded onto the cache or data partition of the relevant memory in the accused LGE mobile product. Ancora Final Infringement Contentions, Ex. B, pp. 93-100. Ancora also alleges that the updates are delivered to accused LGE mobile products via a distribution system that includes various non-LGE servers, including those operated by third party CDNs. Ancora Final Infringement Contentions, Ex. B, pp. 97-98.

328. Based on Ancora's assertions regarding the scope of this limitation, the Arbaugh Patent discloses this limitation. The Arbaugh Patent discloses that when verification of a component fails, it may be repaired by obtaining it from a trusted source and replaced:

> The recovery code contacts a "trusted" host through the AEGIS recovery protocol, discussed below, to recover a signed copy of the failed component. The failed component is then shadowed or repaired, and the system is restarted (warm boot).

Arbaugh Patent, 11:3-8; Arbaugh Provisional at 17:11-14. The network-based recovery process is invoked whenever a component other than BIOS 112 fails:

> If the component that fails its integrity check is a portion of BIOS 112, then it must be recovered from AEGIS ROM 256. The recovery process is a simple

> memory copy from the address space of AEGIS ROM 256 to the memory address of the failed component, in effect shadowing the failed component. A failure beyond BIOS 112 causes the system to boot into a recovery kernel contained on AEGIS ROM 256.

Arbaugh Patent, 10:56-63; Arbaugh Provisional, 16:30-17:5.

329. As discussed earlier, the replacing component includes a component signature certificate that is embedded in the component. Arbaugh Patent, 14:33-45; Arbaugh Provisional, 23:29-30:13. The certificate contains a digital signature of the component. *Id.* Thus, in the network-based recovery process, the encrypted license record, *i.e.*, digital signature of the component (and encrypted hash value which is part of the digital signature), is transferred from the host to the computer.

5. __Limitation 3[e] is Disclosed by the Arbaugh Patent__

330. Limitation 3[e] of the '941 patent recites:

> storing the encrypted license record in the erasable non-volatile memory area of the BIOS.

331. I understand that Ancora contends that this limitation can be met by downloading the accused license record into a "cache" or "data" partition of a device. Ancora Final Infringement Contentions Ex. B, at 93-100. Based on my review of those contentions, Ancora appears to assert, in a conclusory manner, that the "cache" or "data" partition of the memory of accused LGE devices are part of the "BIOS." While Ancora does not explicitly state why it contends that the "cache" or "data" portions of the memory are necessarily part of the "BIOS," it may be based on its interpretation of the Court's construction of BIOS, which is:

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

Dkt. 69, Final Claim Constructions of the Court, at 2. Ancora's interpretation appears to be that any memory constitutes part of the BIOS if it contains code that is "run automatically when a computer is turned on" if it has some role in "start[ing] the operating system" or "support[ing] the transfer of data among hardware devices," even if the memory is not part of what is typically considered a BIOS by a person having ordinary skill in the art.[9]

332. Based on Ancora's assertions regarding the scope of this limitation, the Arbaugh Patent discloses this limitation. The Arbaugh Patent discloses that when programs are retrieved from a network host, they are placed in memory that is used by the BIOS when it is starting up the computer, testing hardware, and starting the operating system, which are all indicia of what constitutes the "BIOS" under the Court's construction.

333. The Arbaugh Patent discloses that its recovery process replaces programs and certificates already stored on the workstation:

> The recovery code contacts a "trusted" host through the AEGIS recovery protocol, discussed below, to recover a signed copy of the failed component. The failed component is then shadowed or repaired, and the system is restarted (warm boot).

Arbaugh Patent, 11:3-8; Arbaugh Provisional, 17:11-14. The network-based recovery process is invoked whenever a component other than BIOS 112 fails:

> If the component that fails its integrity check is a portion of BIOS 112, then it must be recovered from AEGIS ROM 256. The recovery process is a simple memory copy from the address space of AEGIS ROM 256 to the memory address of the failed component, in effect shadowing the failed component. A failure beyond BIOS 112 causes the system to boot into a recovery kernel contained on AEGIS ROM 256.

Arbaugh Patent, 10:56-63, Arbaugh Provisional, 16:30-17:5.

[9] There is an apparent inconsistency between the argument that Ancora makes in its infringement contentions, that any memory storing data received that is used by the BIOS can constitute a "memory for the BIOS," and its statements in its POPR regarding the limited circumstances in which a memory can be considered a "memory for the BIOS," see Section VIII.E,above. I address Ancora's POPR arguments later in this section.

334. The Arbaugh Patent discloses that certificates for all components relied on during the boot process are contained in the "lower layer, first layer 200" of BIOS 112:

> The lowest layer, first layer 200 contains the small section of trusted software, digital signatures, public key certificates, and recovery code AEGIS relies on throughout the boot process.

Arbaugh Patent, 8:45-48; Arbaugh Provisional, 13:12-14. Figure 2a shows that this "lowest layer" of the process uses BIOS section 1 202:

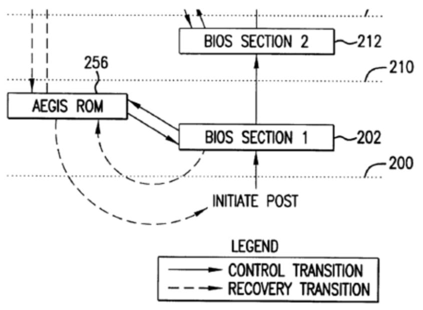


FIG. 2a

Arbaugh Patent, Fig. 2a; Arbaugh Provisional, Fig. 2a. As explained in the Arbaugh Patent, the two sections 202 and 212 of the BIOS 112 are stored in a single flash ROM, which includes one portion which is non-erasable and one portion that is erasable:

> First section 202 and second section 212 can be contained within a single flash ROM, such as the Intel 28FOO1BX-B which has an 8KB block that can be protected from reprogramming while the remainder of the ROM can be reprogrammed. Ideally, first section 202 is stored on this 8KB flash boot block to prevent tampering. Alternatively, an additional ROM card can be used to store the "trusted software", if memory constraints prevent the inclusion of the "trusted Software' within the BIOS ROM.

Arbaugh Patent, 9:17-25; Arbaugh Provisional, 14:8-18 (disclosing dividing BIOS into two logical sections and verification of second BIOS section), 16:30-17:8 (disclosing replacement of a component failing verification by a valid version), 11:11-16 (disclosing use of PROMs for memory components).[10] Notably, a POSITA would have understood the disclosure that the trusted software could be stored somewhere other than the BIOS ROM to indicate that the trusted software is all ordinarily stored in the BIOS ROM of the Arbaugh Patent. Based on this disclosure, a POSITA would have understood that the certificates used during the trusted boot process are stored in the erasable part of the BIOS ROM, and that certificates received during the recovery process that "replace" those certificates would necessarily be stored in the same location in order to be used during the boot process.

335. This conclusion still holds in view of Ancora's argument regarding the scope of the "memory of the BIOS" in its POPR. There, Ancora argued that:

> Absent a showing that the BIOS module actually uses another portion of such physical memory space as part of its normal operations, only the BIOS module would be understood by a person of ordinary skill to be "memory of the BIOS."

336. *Id.*, at 30. Regardless of whether that interpretation is correct, the Arbaugh Patent discloses that the portion of memory that stores the certificates is a memory space that is used by the BIOS module as part of its normal operations. In particular, the Arbaugh Patent discloses that the BIOS code that checks whether any components need to be replaced, step 261 of Figure 2c, is invoked during the boot process. Any updates that occur during this process will necessarily replace the certificates stored for updated components, because certificates for old components will not serve to verify the integrity of new components that have different cryptographic hashes. See Arbaugh Patent, 11:1-14; Arbaugh Patent, 17:9-26.

337. Additionally, any components replaced during the recovery process, step 298 of

[10] Although the Provisional Application does not mention the Intel flash memory chip, chips like that which implemented a protected part and an erasable part were known to those of ordinary skill in the art. In fact the Intel 28FOO1BX-B flash memory chip was marketed since at least 1995. Intel 1-MBIT (128K x 8) BOOT BLOCK FLASH MEMORY, 28F001BX-T/28F001BX-B/28F001BN-T/28F001BN-B (1995); DEFS_ANCORA00001136-DEFS_ANCORA00001169. See footnote 4, above.

Figure 2c, will replace the certificates present in the BIOS. Arbaugh Patent, 20:59-63, Arbaugh Provisional Application, 35:20-23. This replacement process is invoked by the execution of the BIOS, using code present in the BIOS, and uses a location in memory on the same physical chip as the BIOS to store replacement certificates. A POSITA would have therefore understood that the Arbaugh Patent discloses the condition that Ancora asserted in its POPR must be met for memory locations on the BIOS memory chip to be a "memory of the BIOS."

D. Claim 6 is Anticipated by the Arbaugh Patent

338. Claim 6 of the '941 Patent recites:

> 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.

339. The Arbaugh Patent renders claim 6 invalid because it discloses all of the limitations discussed above for claim 1, from which claim 6 depends, and also discloses the step of "establishing a license authentication bureau" in dependent claim 6.

340. As explained earlier in at least limitation 1[a], Arbaugh discloses loading a licensed-software-program into the volatile memory of the computer. A POSITA would have readily understood that the programs in Arbaugh are loaded into volatile memory in order to be executed, as was conventional in the art at the time.

341. Ancora in its infringement contentions has effectively alleged that a hash of a software component is part of the content of that software component. In particular, Ancora alleged that the hash carried in the VB Meta data structure is content of the software component. *See* Ancora Final Infringement Contentions Ex. B, at 100-104. Based on such a broad interpretation of "said licensed-software-program includes contents used to form the license record" as recited in claim 6 (which interpretation I do not agree with), a POSITA would have

understood that the hash of a software component is content of the software component. Arbaugh Patent, 14:33-45, 15:49-67; Arbaugh Provisional, 23:29-24:13, 26:14-25. The Arbaugh Patent discloses that its component certificate, which carries the digital signature, includes the hash in encrypted form within the digital signature. Arbaugh Patent, 14:33-45; Arbaugh Provisional 23:29-24:13. For that reason, if Ancora's asserted claim scope for purposes of infringement is assumed to be correct, then Arbaugh Patent discloses that "said licensed-software-program includes contents used to form the license-record."

E. Claim 7 is Anticipated or Rendered Obvious by the Arbaugh Patent

1. __Limitation 7[a] is Disclosed or Rendered Obvious by the Arbaugh Patent__

342. Limitation 7[a] of the '941 patent recites:

> A method according to claim 6 wherein using an agent to set up the verification structure includes the steps of: establishing or certifying the existence of a pseudo-unique key in a first non- volatile memory area of the computer;"

343. As discussed earlier in connection with claim 1[b], the agent in the Arbaugh Patent, is the software used to implement the AEGIS model. This software establishes or certifies the existence of a pseudo-unique key in a first non-volatile memory area.[11] I have already addressed earlier, for the claim limitation "using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS," how the Arbaugh Patent discloses establishing or certifying the existence of a pseudo-unique key in the non-volatile memory of the BIOS (because that is how the Court construed the term "set up a verification structure.")[12]

344. The further requirement that the pseudo-unique key be in a "first non-volatile

[11] I understand that the term "first non-volatile memory area of the computer" has been given its plain and ordinary meaning by the Court. Dkt. 69 at 5.
[12] The Court construed "set up a verification structure" to mean "establishing or certifying the existence of a pseudo-unique key and establishing at least one license-record location," and I explained above how that limitation is disclosed in the Arbaugh Patent.

memory area of the computer" only requires that the pseudo-unique key be present in one non-volatile memory area. The Arbaugh Patent describes that the cryptographic public key certificates used for its verification process are stored in AEGIS BIOS 112:

> BIOS 112 and AEGIS ROM 256 contain the verification code, and public key certificates. AEGIS ROM 256 also contains code that allows the secure recovery of any integrity failures found during the initial bootstrap.

Arbaugh Patent, 10:49-53; Arbaugh Provisional, 16:26-28. The Arbaugh Patent then clarifies that BIOS 112 is split into "two logical sections" (*i.e.*, implying they may be implemented in the same memory device):

> AEGIS modifies the boot process as shown in FIG. 1a by dividing system BIOS 112 into two logical sections. First section 202 contains the "trusted software", the bare essentials needed for integrity verification and recovery. Second, section 212 contains the remainder of the system BIOS 112 and the CMOS memory.

Arbaugh Patent, 9:12-17; Provisional Application, 14:8-12; see also Fig. 2a, showing "first layer 200" including "BIOS SECTION 1 202." This first section 202 of the BIOS contains "cryptographic code and public key certificates," because later in the same paragraph the Arbaugh Patent allows that "public key certificates" could, instead of being placed in first section 202 of BIOS 112, be placed in a cryptographic coprocessor. Arbaugh Patent, 9:26-32; see also Provisional Application, 14:9-11, 13:12-14. Because this is an alternative only available when a cryptographic coprocessor is present, the Arbaugh Patent clearly discloses that there is at least one public key certificate, *i.e.* pseudo-random key, present in the system BIOS 112, which is a non-volatile memory area of the computer. *See also* Provisional Application, 14:9-11, 13:12-14. 184.

345. In particular, because (i) a component certificate (which carries the digital signature) is embedded within a component (Arbaugh Patent, 14:43; Arbaugh Provisional, 24:12), (ii) the component certificate and component are written to the AEGIS BIOS (*i.e.*, at

least the erasable portion of AEGIS BIOS) when the component fails verification and is replaced from the network or a the AEGIS ROM (Arbaugh Patent, 9:11-39 and 10:47-11:15; Arbaugh Provisional, 14:8-18 and 16:23-17:8), and (iii) the component certificate includes both the digital signature of the component and the public key (Arbaugh Patent, 14:33-45; Arbaugh Provisional 23:29-24:13). An example of an AEGIS component certificate is shown below:

```
((cert (issuer (hash-of-key (hash sha1 approverkey)))
        (Subject (hash sha 1 hashtytes))
        (not-before 09/01/97-0000)
        (not-after 09/05/97-0000))
(signature (hash sha 1 hashbytes)
        (public-key disa-sha1 approverkey)
(Sigbytes)))
```

Arbaugh Patent, 14:34-42; Arbaugh Provisional, 24:1-11. The component certificates represent the digitally-signed cryptographic hashes of each program, and are stored in the erasable portion of the non-volatile memory that stores the BIOS because they can be downloaded and replaced in the AEGIS system. (*i.e.*, the erasable part of the AEGIS BIOS.)

346. This limitation is explicitly disclosed by the Arbaugh Patent under Ancora's interpretation of the claims as disclosed in Ancora's Final Infringement contentions. Even if it were not, this limitation would have been rendered obvious to a POSITA in view of the Arbaugh Patent. The Arbaugh Patent discloses that a public key is present in the BIOS of the computer system. The Court's construction requires "establishing" or "certifying" the existence of the public key in a non-volatile memory area of the computer. It would have been obvious to a POSITA that in order for the system in the Arbaugh Patent to operate the public key must be present in the non-volatile memory of the computer.

347. This would be apparent because the public key is needed to perform verifications at the time the computer is powered on, before any data can be received from an outside source.

As such, the public key must be present, or be "established," prior to execution of the integrity verification process described in Arbaugh. Additionally, a POSITA would have understood that the process described in the Arbaugh Patent would not work if the public key were not present, because there would be no way to verify the signature of any components, and operation would thus never be permitted. As such, it would have been evident to a POSITA that the Arbaugh Patent would explicitly certify, or detect and verify, the presence of the public key in the memory as part of its operation.

348. Under the claim scope that Ancora appears to afford to this claim limitation, it is disclosed by either the client software or the server software disclosed in the Arbaugh Patent, working independently or together, for the reasons described herein as well as in limitation 1[b], above.

2. <u>Limitation 7[b] is Disclosed by the Arbaugh Patent</u>

349. Limitation 7[b] of the '941 patent recites:

> establishing at least one license-record location in the first nonvolatile memory area or in the erasable, nonvolatile memory area of the BIOS.

350. The Arbaugh Patent also discloses establishing at least one license-record location in the first nonvolatile memory area or in the erasable, nonvolatile memory area of the BIOS.

351. I understand that the Court has indicated that "establishing at least one license-record location" may include "forming a license-record by at least partially encrypting the contents used to form a license-record with other predetermined data contents, using at least part of the pseudo-unique key; and storing the encrypted license-record." I further understand the Court indicated that this information is not for the jury to consider. See Dkt. 69 at 4. However, I

352. understand that because the Court has determined as a matter of law that the claim scope includes such an operation, that evidence that the Arbaugh Patent discloses that operation

shows that the Arbaugh Patent discloses "establishing at least one license record location."

353. As described earlier, the cryptographic hash in the Arbaugh Patent is generated by taking the bits in a computer program, and using an algorithm like SHA1 (described in the Arbaugh Patent) to generate a cryptographic hash. This cryptographic hash is then encrypted, by using the pseudo-unique encryption key, to generate a signature. Arbaugh describes the use of public key cryptography, which I described above, in order to ensure the integrity of the BIOS components:

> the integrity of the bootstrap process and provides reliability. Integrity is validated at each layer transition in the bootstrap process and a recovery process is included for integrity check failures. Ensuring the integrity is provided by the use of public key cryptography, a cryptographic hash function, and public key certificates.

Arbaugh Patent, 4:34-40; Arbaugh Provisional, 7:4-7. Public key cryptography was known to people having ordinary skill in the art at the time as a way of authenticating a message, that is, confirming that a message originated from whomever claims to have sent it. It was also known that this technique could be used to verify the source identity and the integrity of a file by "signing" a cryptographic hash. Arbaugh Patent, 15:17-67; Arbaugh Provisional, 25:17-26:25. This is done by having the entity which is trusted to store execution rights, which in the '941 Patent is the license authentication bureau, and in the Arbaugh Patent is the trusted repository, use its cryptographic key to encrypt the cryptographic hash to yield a digital signature. A POSITA would have known encryption techniques like public key cryptography can be used to digitally sign the hash. In this approach the source of the execution rights for the program in question derive a cryptographic hash using any of a number of well-known techniques like MD5, then uses its private key to generate an encrypted version of the cryptographic hash. This step is important because anyone that is able to decrypt the encrypted hash can (1) be assured of the source's identity then (2) compare the recovered (decrypted) cryptographic hash to its own calculated cryptographic hash and, If they match be assured of the program file integrity.

354. The Arbaugh Patent uses public key according to the Digital Signature Standard, DSS. Arbaugh Patent, at 15:18-48; Arbaugh Provisional, at 25:17-26:13 which I described above. As explained in the Arbaugh Patent, and would have been known by a person having ordinary skill in the art, DSS applies a digital signature algorithm (DSA) having a private key (x) and a public key (y). Arbaugh Patent, at 15:18-22, 32-33; Arbaugh Provisional, 25:17-21, 26:1-2. The private key is used during signature generation, and the public key is used during signature verification, as I explained earlier in my background discussion of DSS.

355. Based on Ancora's infringement contentions, I understand Ancora has asserted that the public key in a public key cryptographic system meets the limitation of being a "pseudo-unique key." The digital signature in Arbaugh is generated by a signing operation is stored in the erasable, non- volatile memory area of the BIOS. The Arbaugh Patent describes storing digital signatures in the device that has the AEGIS BIOS. Arbaugh Patent, 4:43-45; 8:45-48 ("first layer 200 contains the small section of trusted software, digital signatures, public key certificates, and recovery code AEGIS relies on throughout the boot process."); Arbaugh Provisional, 7:10-11, 13:12-14. The signature is stored twice in the use of the AEGIS model as described in the Arbaugh Patent. It is stored a first time when the system is first configured, since the AEGIS model assumes the presence of the signatures when the ROM is first used in the device. *Id.* In order to be present in the AEGIS BIOS the signature must first be stored there.

356. The signature is also stored in the operation of the AEGIS model when the system determines that a component needs to be replaced with a known-good version from the trusted repository:

> Where network host 254 is the trusted repository, the detection of an integrity failure causes the system to boot into a recovery code contained on the AEGIS

> ROM 256. The recovery code contacts a “trusted' host through the AEGIS recovery protocol, discussed below, to recover a signed copy of the failed component. The failed component is then shadowed or repaired, and the system is restarted (warm boot).

Arbaugh Patent, 11:1-8; Arbaugh Provisional, 17-9-14. Periodically, the certificate (and by extension the signature) can expire, in which case the AEGIS model will download a new version of the certificate alone, which meets this limitation. Arbaugh Patent, 5:7-12; Arbaugh Provisional, 8:4-7. Alternatively, if the component itself needs to be updated, then both the program and the certificate are downloaded. *Id.* The certificates are stored in the BIOS 112, which is a non-volatile memory, and because the certificates must be replaceable that non-volatile memory must be erasable. This limitation is therefore disclosed by the Arbaugh Patent.

357. Under the claim scope that Ancora appears to afford to this claim limitation, it is disclosed by either the client software or the server software disclosed in the Arbaugh Patent, working independently or together, for the reasons described herein as well as in limitation 1[b], above.

F. Claim 8 is Anticipated by the Arbaugh Patent

1. Limitation 8[a] is Disclosed by the Arbaugh Patent

358. Limitation 8[a] of the ’941 patent recites:

> A method according to claim 6 wherein establishing a license- record includes the steps of: forming a license-record by encrypting of the contents used to form a license-record with other predetermined data contents, using the key

359. For the preceding limitation I explained why the limitation was met under the Court’s note that “establishing at least one license-record location” may include “forming a license-record by at least partially encrypting the contents used to form a license-record with other predetermined data contents, using at least part of the pseudo-unique key; and storing the encrypted license-record.” I then explained why the Arbaugh Patent discloses that specific

circumstances. Because this element is entirely subsumed by the analysis above for claim 7, this element is disclosed by the Arbaugh Patent for the same reasons as claim 7.

360. Under the claim scope that Ancora appears to afford to this claim limitation, it is disclosed by either the client software or the server software disclosed in the Arbaugh Patent, working independently or together, for the reasons described herein as well as in limitation 1[b], above.

2. Limitation 8[b] is Disclosed by the Arbaugh Patent

361. Limitation 8[b] of the '941 patent recites:

> establishing the encrypted license-record in one of the at least one established license-record locations.

362. The encrypted license record identified above is stored in one of the established license-record locations. As discussed earlier for claim 7, the license-record location is the location where the license-record is stored. In the Arbaugh Patent, the license-record is stored in AEGIS ROM 256 in the location dedicated to storing verification codes and public key certificates. Arbaugh Patent, at 10:47-67; Arbaugh Provisional, 16:23-17:8.

363. Under the claim scope that Ancora appears to afford to this claim limitation, it is disclosed by either the client software or the server software disclosed in the Arbaugh Patent, working independently or together, for the reasons described herein as well as in limitation 1[b], above.

G. Claim 9 is Anticipated by the Arbaugh Patent

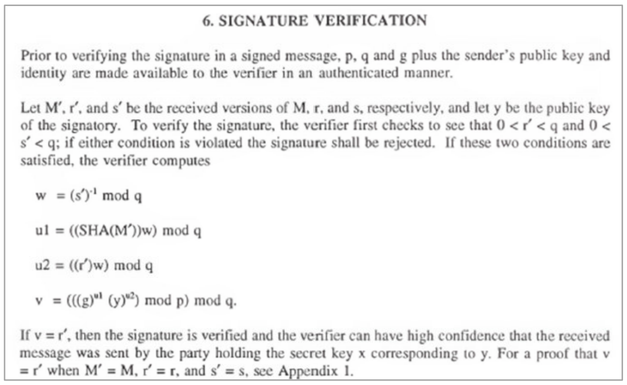
364. The Arbaugh Patent renders claim 9 invalid because it discloses all of the limitations discussed above for claim 7, from which claim 9 depends, and also discloses the additional limitations in dependent claim 9.

1. Limitation 9[a] is Disclosed by the Arbaugh Patent

365. Limitation 9[a] of the ’941 patent recites:

> A method according to claim 7 wherein verifying the program includes the steps of: encrypting the licensed-software-program's license-record contents from the volatile memory area or decrypting the license-record in the erasable, nonvolatile memory area of the BIOS, using the pseudo-unique key

366. The Arbaugh Patent discloses this limitation at least in its disclosure of verifying the digital signature of the cryptographic hash in component certificates, which constitutes “decrypting the license-record in the erasable, nonvolatile memory area of the BIOS, using the pseudo-unique key.” The Arbaugh Patent discloses using DSS to verify sender’s identity and the the integrity of components, by comparing the cryptographic hash of a program to be verified against the digital signature received from the trusted repository. Arbaugh Patent, 6:10-16, 15:18-48 and 11:1-6; Arbaugh Provisional 7:7-11, 25:17-26:13 and 17:9-13. DSS describes its verification procedure as shown to the right (DSS at 7). A POSITA would have understood that the “message” M and M’ in the case of the Arbaugh Patent is the program to be verified, and r and s are the digital signature. See DSS at 6 (“The signature of a message M is the pair of numbers r and s . . .”).[13] The verification process includes a comparison between v, a computed



6. SIGNATURE VERIFICATION

Prior to verifying the signature in a signed message, p, q and g plus the sender's public key and identity are made available to the verifier in an authenticated manner.

Let M′, r′, and s′ be the received versions of M, r, and s, respectively, and let y be the public key of the signatory. To verify the signature, the verifier first checks to see that $0 < r' < q$ and $0 < s' < q$; if either condition is violated the signature shall be rejected. If these two conditions are satisfied, the verifier computes

$$w = (s')^{-1} \bmod q$$

$$u1 = ((SHA(M'))w) \bmod q$$

$$u2 = ((r')w) \bmod q$$

$$v = (((g)^{u1} (y)^{u2}) \bmod p) \bmod q.$$

If $v = r'$, then the signature is verified and the verifier can have high confidence that the received message was sent by the party holding the secret key x corresponding to y. For a proof that $v = r'$ when $M' = M$, $r' = r$, and $s' = s$, see Appendix 1.

[13] As an alternative, of course, the DSS verification algorithm can be considered to operate on the value of SHA(M), which is simply the cryptographic hash of the program bits. Either way, the signature is on the cryptographic hash, whether the cryptographic hash value is stored or is derived algorithmically when needed.

result, and r', which is part of the signature. A POSITA would have understood that this constitutes "decrypting" the signature because it involves inputting, in the first instance, s' as an input, which is then run through processing steps involving the use of the public key y. The conversion of one set of data (here, s') to another set of data (here, v) using a public key constitutes "decryption."

367. The Arbaugh Patent discloses that the decryption is performed using the public key of the trusted repository, y, as described above. I understand based on Ancora's infringement contentions, they have asserted that the public key in a public key cryptographic system meets the limitation of being a "unique key." To the extent that assertion is assumed to be correct, this limitation is disclosed by the Arbaugh Patent because it also discloses a public key used to perform decryption of a digitally signed cryptographic hash to verify the source identity and the integrity of a component.

368. Assuming the claim scope is as broad as Ancora asserts, the Arbaugh Patent also discloses that the decryption is of "the license record data accommodated in the erasable second non-volatile memory area" because it discloses that integrity verification is performed by using the component certificate, which includes both the cryptographic hash and a digital signature of the hash. The component certificate includes the digitally signed cryptographic hash and a copy of the public key:

```
((cert (issuer (hash-of-key (hash sha1 approverkey)))
        (subject (hash sh1 hashtytes))
        (not-before 09/01/97-0000)
        (not-after 09/06/97-0000)
(signature (hash sha1 hashbytes)
        (public-key dsa-sha1 approverkey)
(sigbytes)))
```

Arbaugh Patent, 14:33-45; Arbaugh Provisional, 23:29-24:13. Further, the certificate is stored in the erasable non-volatile memory area because, as described above for claim 1, the certificate is stored as part of the AEGIS BIOS and can be replaced. Arbaugh Patent, 10:47-11:8; Arbaugh Provisional, 16:23-17:14. In order to be replaceable, the portion of the memory in which the certificate is stored must be erasable. This limitation is therefore disclosed by the Arbaugh Patent.

2. ***Limitation 9[b] is Disclosed by the Arbaugh Patent***

369. Limitation 9[b] of the '941 patent recites:

> comparing the encrypted licenses-software-program's license- record contents with the encrypted license-record in the erasable, nonvolatile 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.

370. Ancora in its infringement contentions has effectively alleged that a hash of a software component is part of the content of that software component. See Ancora Final Infringement Contentions Ex. B, at 130-142. In particular, Ancora alleged that the hash carried in the VB Meta data structure is content of the software component. *Id.* Based on such a broad interpretation of "license-software-program's license record contents" as recited in claim 9 (which interpretation I do not agree with), a POSITA would have understood that the hash of a software component is content of the software component, which satisfies the claim element.

371. Under this broad interpretation, the Arbaugh Patent discloses this limitation at least in its disclosure of verifying the digital signature of the cryptographic hash in component certificates. This constitutes "comparing the license-software program's license-record contents with the decrypted license-record in erasable non-volatile memory area of the BIOS." The Arbaugh Patent describes the verification process as including a comparison between a computed cryptographic hash value and a stored digital signature. Arbaugh Patent, at 4:43-45 ("The integrity checks compare a computed cryptographic hash value with a stored digital signature associated with each component"); Arbaugh Provisional, at 7:10-11.

372. Assuming Ancora's asserted claim scope is correct, in the Arbaugh Patent, the cryptographic hash is used to "verify" the "licensed program" because it is one of the two pieces of data compared in the AEGIS model to verify the integrity of the program component. The second piece of data used for the comparison in the AEGIS model is the cryptographically signed cryptographic hash provided by the trusted repository (or other authority responsible for certifying the integrity of a program component). Arbaugh Patent 11:1-15 and 14:33-45;

373. Arbaugh Provisional 17:9-19 and 23:29-24:13. This information is contained in the component certificate:

```
((cert (issuer (hash-of-key (hash sha1 approverkey)))
        (subject (hash sh1 hashtytes))
        (not-before 09/01/97-0000)
        (not-after 09/06/97-0000)
(signature (hash sha1 hashbytes)
        (public-key dsa-sha1 approverkey)
(sigbytes)))
```

Arbaugh Patent, 14:33-45, 15:49-67; Arbaugh Provisional, 23:29-24:13, 26:14-25. Here "hashbytes" is the hash value derived using SHA1.

374. DSS is used to perform a comparison between the cryptographic hash of the program and the digital signature received from the trusted repository. Arbaugh Patent, 6:10-16, 15:18-48 and 11:1-6; Arbaugh Provisional 7:7-11, 25:17-26:13 and 17:9-13. The procedure used to verify the cryptographic hash was described for the preceding element, and involves decrypting the digital signature to extract the cryptographic hash for purposes of the verification.

H. Claim 10 is Anticipated by the Arbaugh Patent

375. Limitation 10 of the '941 patent recites:

> 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.

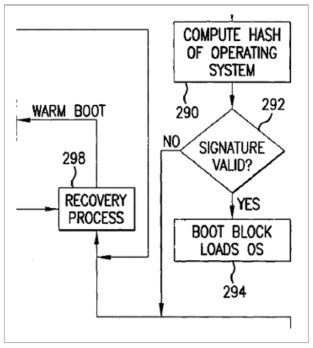
376. The Arbaugh Patent renders claim 10 invalid because it discloses all of the

limitations discussed above for claim 9, from which claim 10 depends, and also discloses that "acting on the program includes the step: restricting the program's operation with predetermined limitations if the comparing yields non-unity or insufficiency" in dependent claim 10.

377. The Arbaugh Patent discloses that when the integrity of a program cannot be verified, there are several predetermined limitations that can be applied to the execution:

> The AEGIS integrity policy prevents the execution of a component if its integrity cannot be validated. There are three reasons why the integrity of a component could become inval*id.* The integrity of the component could change because of some hardware or software malfunction, the integrity of the component could change because of some malicious act, or the component's certificate time stamp may no longer be val*id.* In each case, AEGIS attempts to recover from a trusted repository, step 298, as discussed below.

Arbaugh Patent, at 10:13-19; see also *id.* at 1:32-41; Arbaugh Provisional, 15:19-25; see also *id.* at 1:21-2:4. A POSITA would have understood from this disclosure that the AEGIS model verifies integrity of the program. If the program integrity is not verified, then program execution halts while the AEGIS system attempts to recover from the trusted repository. In this case, the program resident on the computer is not executed, which is a "predetermined limitation" because the system is explicitly configured to prevent the program from executing while the trusted repository recovery proceeds. A person of ordinary skill in the art would understand this to be required based on the disclosure in the Arbaugh Patent that after the trusted repository recovery process, the failed component is restored from the trusted repository and the system is then restarted. Arbaugh Patent, 10:56-67; Arbaugh Provisional, 16:30-17:5. This is shown in several places in the Arbaugh Patent, including in the flow chart in Figures 2b and 2c where, prior to loading the operating system, the signature must be confirmed as valid*.* If the signature is not valid, then execution does not continue to load the operating system, but instead branches to the recovery process, as shown in the excerpt below:



Arbaugh Patent, Figs. 2b, 2c. Arbaugh Provisional, Figs. 2b, 2c.

378. The Arbaugh Patent then discloses an alternative path if the recovery process does not complete successfully:

> Should a trusted repository be unavailable after several attempts, then the client's further action depends on the security policy of the user. For instance, a user may choose to continue operation in a limited manner or may choose to halt operations altogether.

Arbaugh Patent, 10:20-25, Arbaugh Provisional, 15:26-16:2. This passage describes the "predetermined limitations" that are applied in the Arbaugh Patent if integrity is not verified, that is, if the comparison yields an insufficiency. According to the Arbaugh Patent there is a security policy set by the user which determines what occurs if the trusted repository cannot be contacted. A security policy must, by definition, be predetermined because a policy is a rule set in place to adjudicate how to resolve a later-occurring problem. The two choices of execution path given by the Arbaugh Patent are continued operation in a limited manner, which constitutes a predetermined limitation, or the halt of operations, which is also a predetermined limitation.

I. Claim 11 is Anticipated by the Arbaugh Patent

379. Claim 11 of the '941 patent recites:

> A method according to claim 1 wherein the volatile memory is a RAM.

380. The Arbaugh Patent renders claim 11 invalid because it discloses all of the limitations discussed above for claim 1, from which claim 11 depends, and also discloses that "the volatile memory is a RAM" in dependent claim 10.

381. The Arbaugh Patent explains that the computer implementing the AEGIS model includes a RAM memory. Arbaugh Patent, 6:49-51 ("Computer System 1 also includes a main memory 8, preferably random access memory (RAM) and a ROM BIOS 2, which stores the system BIOS."), Arbaugh Provisional, 10:25-11:2 (disclosing implementation of AEGIS on practical, real-world computer systems such as those based on the IBM PC architecture). This limitation is therefore met, as the main memory 8 is the volatile memory relied on in claim 1.

J. Claim 12 is Anticipated or Rendered Obvious by the Arbaugh Patent

382. Claim 12 of the '941 patent recites:

> The method of claim 1, wherein a pseudo-unique key is stored in the non-volatile memory of the BIOS.

383. The Arbaugh Patent renders claim 12 invalid because it discloses all of the limitations discussed above for claim 1, from which claim 12 depends, and also discloses that "a pseudo-unique key is stored in the non-volatile memory of the BIOS" in dependent claim 12.

384. The Arbaugh Patent discloses storing a public key of the trusted repository, as I explained above for limitation 1[b]. I understand based on Ancora's infringement contentions, they have asserted that the public key in a public key cryptographic system meets the limitation of being a "pseudo-unique key." To the extent that assertion is assumed to be correct, this limitation is disclosed by the Arbaugh Patent.

385. The Arbaugh Patent discloses a public key used to verify a digital signature on cryptographic hashes provided by the trusted repository. Those keys are stored in the AEGIS BIOS 112, which is the non-volatile memory of the BIOS:

> BIOS 112 and AEGIS ROM 256 contain the verification code, and public key certificates. AEGIS ROM 256 also contains code that allows the secure recovery of any integrity failures found during the initial bootstrap.

Arbaugh Patent, 10:49-53; Arbaugh Provisional, 16:26-28.

386. Additionally, the Arbaugh Patent discloses that the public key used to verify each component is included as part of the component certificates received from the trusted authority to verify components:

> An example of an AEGIS component certificate is shown below:
>
> ```
> ((cert (issuer (hash-of-key (hash sha1 approverkey)))
>           (Subject (hash sha 1 hashtytes))
>           (not-before 09/01/97-0000)
>           (not-after 09/05/97-0000))
> (signature (hash sha 1 hashbytes)
>           (public-key disa-sha1 approverkey)
> (Sigbytes)))
> ```

Arbaugh Patent, 14:34-42; Arbaugh Provisional, 24:1-11. As described earlier, the component certificates represent the digitally-signed cryptographic hashes of each program, and are stored in the erasable portion of the non-volatile memory because they can be downloaded and replaced in the AEGIS system. See claim 1[b], supra. Because Ancora contends that a public key stored in the memory of the computer meets this limitation, under that claim scope, this limitation is disclosed by the Arbaugh Patent.

387. This limitation is explicitly disclosed by the Arbaugh Patent under Ancora's interpretation of the claims as is evident from Ancora's Final Infringement Contentions. Even if it were not, this limitation would have been rendered obvious to a POSITA in view of the Arbaugh Patent. The Arbaugh Patent discloses, in a portion found verbatim in the Arbaugh Provisional, that public keys are stored in its BIOS. Arbaugh Patent, 8:45-48; Arbaugh

388. Provisional, 13:13-18. The Arbaugh Patent and Provisional also both disclose that public key component certificates issued by a private key holder may be issued with expiration

periods:

```
((cert (issuer (hash-of-key (hash shal Serverkey)))
        (Subject (hash-of-key (hash sha 1 serverkey)))
        (tag (server (dh-ggbytes)
                (dh-p pbytes)
                (dh-Y ybytes)
                (msg-hash
                (hash sha1 hbytes))
                (cnonce cbytes)
                (Snonce sbytes)))
        (not-before 09/01/97-0900)
        (not-after 09/01/97-0900)) (signature
        (hash shal hashbytes)
        (public-key disa-sha1 serverkey)
(Sigbytes)))
```

Arbaugh Patent, 13:40-14:42; Arbaugh Provisional, 21:34-23:28. A POSITA would have recognized that because the certificates authenticating public keys can expire, they may in the ordinary course of operation would need to be replaced. A POSITA would have therefore been motivated to configure the system in such a way that public key certificates could be updated without requiring a BIOS ROM or motherboard to be replaced, and accordingly, placing public key certificates in an erasable portion of the ROM BIOS. Such a modification would have been well within the abilities of a POSITA and would not have required undue experimentation.

389. This conclusion still holds in view of Ancora's argument regarding the scope of the "memory of the BIOS" in its POPR for substantially the same reasons I have described as for limitation 1[preamble] and 1[b], above.

K. Claim 13 is Anticipated by the Arbaugh Patent

390. Claim 13 of the '941 patent recites:

> The method of claim 1, wherein a unique key is stored in a first non-volatile memory area of the computer.

391. The Arbaugh Patent renders claim 13 invalid because it discloses all of the limitations discussed above for claim 1, from which claim 12 depends, and also discloses that "a

unique key is stored in a first nonvolatile memory area of the computer" in dependent claim 12.

392. The Arbaugh Patent discloses storing a public key of the trusted repository, as I explained above for claim 12 and limitation 1[b], which Ancora contends in its infringement contentions meets the limitation of being a "unique key." To the extent that assertion is assumed to be correct, this limitation is disclosed by the Arbaugh Patent.

393. The public key in the Arbaugh Patent is stored in a nonvolatile memory area of the computer, as described above, because the AEGIS BIOS is a nonvolatile memory.

L. Claim 14 is Anticipated by the Arbaugh Patent

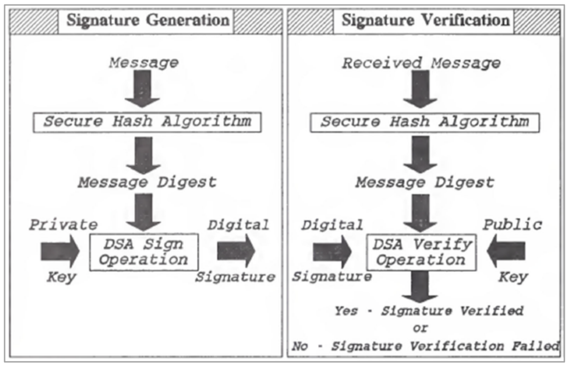
394. Claim 14 of the '941 patent recites:

> 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.

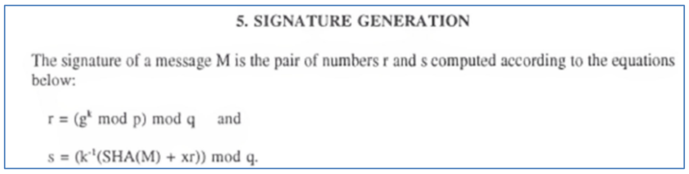
395. The Arbaugh Patent renders claim 14 invalid because it discloses all of the limitations discussed above for claim 13, from which claim 14 depends, and also discloses that "the step of using the agent to set up the verification record, including the license record, includes encrypting a license record data in the program using at least the unique key" in dependent claim 14.

396. Ancora in its infringement contentions has effectively alleged that a hash of a software component is part of the content of that software component. *See* Ancora Final Infringement Contentions Ex. B, at 153-161. In particular, Ancora alleged that the hash carried in the VB Meta data structure is content of the software component. *Id.* Based on such a broad interpretation of "encrypting a license record data in the program" as recited in claim 14 (which interpretation I do not agree with), a POSITA would have understood that the hash of a software component is content of the software component.

397. The Arbaugh Patent discloses verification information in the form of a digitally signed cryptographic hash according to the digital signature standard, DSS, as described above in claim 1. In particular, the digital signature included in the Arbaugh Patent's component certificate is formed by taking the cryptographic hash of the program bits and then digitally signing them using the trusted repository's private key, as described in the DSS standard:



398. DSS at 2. The left box, "signature generation," shows taking the "message" and calculating a message digest using the secure hash algorithm, SHA. *Id.* at 5. This is then encrypted using the private key:



5. SIGNATURE GENERATION

The signature of a message M is the pair of numbers r and s computed according to the equations below:

$r = (g^k \bmod p) \bmod q$ and

$s = (k^{-1}(SHA(M) + xr)) \bmod q.$

DSS at 6. In this equation, x is the private key. *Id.*

399. Thus, under Ancora's broad construction of "license record data in the program", the Arbaugh Patent discloses setting up the verification record, including "encrypting a license record data in the program using at least the unique key."

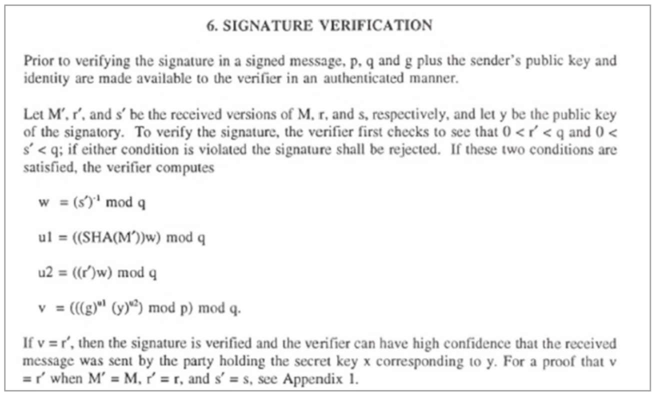
M. Claim 16 is Anticipated by the Arbaugh Patent

400. Claim 16 of the ’941 patent recites:

> 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.

401. The Arbaugh Patent renders claim 16 invalid because it discloses all of the limitations discussed above for claim 13, from which claim 16 depends, and also discloses “decrypting the license record data accommodated in the erasable second non-volatile memory area of the BIOS using at least the unique key” in dependent claim 16.

402. The Arbaugh Patent discloses using DSS to verify the integrity of components, by comparing the cryptographic hash of a program to be verified against the digital signature received from the trusted repository. Arbaugh Patent, 15:17-67; Arbaugh Provisional, 25:17-26:25. DSS describes its verification procedure as follows:



6. SIGNATURE VERIFICATION

Prior to verifying the signature in a signed message, p, q and g plus the sender's public key and identity are made available to the verifier in an authenticated manner.

Let M′, r′, and s′ be the received versions of M, r, and s, respectively, and let y be the public key of the signatory. To verify the signature, the verifier first checks to see that $0 < r' < q$ and $0 < s' < q$; if either condition is violated the signature shall be rejected. If these two conditions are satisfied, the verifier computes

$$w = (s')^{-1} \bmod q$$

$$u1 = ((SHA(M'))w) \bmod q$$

$$u2 = ((r')w) \bmod q$$

$$v = (((g)^{u1} (y)^{u2}) \bmod p) \bmod q.$$

If $v = r'$, then the signature is verified and the verifier can have high confidence that the received message was sent by the party holding the secret key x corresponding to y. For a proof that $v = r'$ when $M' = M$, $r' = r$, and $s' = s$, see Appendix 1.

DSS at 7. A POSITA would have understood that the “message” M and M’ in the case of the Arbaugh Patent is the program to be verified, and r and s are the digital signature. See DSS at 6

("The signature of a message M is the pair of numbers r and s . . .").[14] The verification process includes a comparison between v, a computed result, and r', which is part of the signature. A POSITA would have understood that this constitutes "decrypting" the signature because it involves inputting, in the first instance, s' as an input, which is then run through processing steps involving the use of the public key y. The conversion of one set of data (here, s') to another set of data (here, v) using a public key constitutes "decryption."

403. I understand based on Ancora's infringement contentions that Ancora asserts the public key in a public key cryptographic system meets the limitation of being a "unique key" and that this limitation is met by performing decryption using that public key. To the extent that assertion is assumed to be correct, this limitation is disclosed by the Arbaugh Patent. The Arbaugh Patent discloses using the public key of the trusted repository, y, to perform decryption as described above.

404. I understand based on Ancora's infringement contentions that Ancora asserts that a digitally-signed cryptographic hash is a "license record." The Arbaugh Patent discloses performing integrity verification using a digitally-signed cryptographic hash in the form of a component certificate, which includes both the cryptographic hash and a digital signature of the hash. The component certificate includes the claimed license record data:

```
((cert (issuer (hash-of-key (hash sha1 approverkey)))
        (subject (hash sh1 hashtytes))
        (not-before 09/01/97-0000)
        (not-after 09/06/97-0000)
(signature (hash sha1 hashbytes)
        (public-key dsa-sha1 approverkey)
(sigbytes)))
```

[14] As an alternative, of course, the DSS verification algorithm can be considered to operate on the value of SHA(M), which is simply the cryptographic hash of the program bits. Either way, the signature is on the cryptographic hash, whether the cryptographic hash value is stored or is derived algorithmically when needed.

Arbaugh Patent, 14:33-45; Arbaugh Provisional, 23:29-24:13. Further, the certificate is stored in the erasable non-volatile memory area because, as described above for claim 1, the certificate is stored as part of AEGIS BIOS and can be replaced. Arbaugh Patent, 10:47-11:8; Arbaugh Provisional, 16:23-17:14. In order to be replaceable, the portion of the memory in which the certificate is stored must be erasable. This limitation is therefore disclosed by the Arbaugh Patent.

405. I note that the private key used to generate the digital signature is not the key stored in the memory in the Arbaugh Patent. Instead, it is the public key portion of the asymmetric key pair (private key and public key) which is stored in the memory in the Arbaugh Patent. Ancora contends that the limitation of "encrypting a license record data in the program using at least the unique key" is met by a public key stored on an accused device. See Ancora Final Infringement Contentions, Ex. B, at 161-173.

XIV. OVERVIEW OF THE JABLON PATENT

406. U.S. Patent No. 5,421,006 to Jablon, et al. (the "Jablon Patent") describes a method and device to prevent the execution of corrupted programs, a concept referred to in the Jablon Patent as "integrity." Jablon Patent, Abstract. While the term "corrupted" may suggest a program that has become inadvertently changed, the Jablon Patent is clear that this includes efforts made by an attacker to interpose program code that should not be allowed to execute. *Id.*, at 4:41-43. The Jablon Patent confirms integrity by using a code that is used to verify that the program actually present on a computer is the program that is authorized to run. Jablon calls this code a "modification detection code," or "MDC." *Id.*, *id.* at 5:26-52.

407. The "modification detection code" in the Jablon Patent is a numerical code that is generated by applying mathematical operations to the actual bits of a program, similar to the cryptographic hash in the Arbaugh Patent:

> Modification detection codes are also known by other names in the literature, including: “cryptographic checksum”, “cryptographic hash”, and “message digest”.

Id., at 5:44-47. Like the cryptographic hashes in the Arbaugh Patent, Jablon’s modification detection codes serve as a unique identifier that indicates whether the program is authorized to run. When a program is loaded for execution, the system will calculate a cryptographic hash—or modification detection code—of the program in memory. It will then compare that modification detection code to the stored modification detection code to confirm that they are the same:

> In one variation, modification detection codes provide a suitably strong means of integrity verification. Before the first program transfers control to the second program, it computes, using known techniques, a modification detection code for the second program, and compares it to a pre-computed stored code in protectable non- volatile memory. If the codes are not identical, then the second program is not run, and the user is warned of the problem.

Id. at 8:60-68.

408. A further layer of protection imposed by the Jablon Patent is the use of public-key cryptographic signing of the modification detection code stored in memory, to ensure that the modification detection code itself has not been tampered with. In this embodiment, the modification detection code is digitally signed by a trusted authority and included with the program itself:

> Modification detection codes are used here too, but the code for the second program resides in the same memory device as the program itself. It is protected from malicious modification by being signed with the private key of the trusted authority.

Id. at 9:11 In this embodiment, when verifying the modification detection code, the system will first use a public key stored in the BIOS to confirm that the digital signature on the modification detection code is authenticated:

> The verification process occurs in two steps: first the signed modification detection code is retrieved during the “unsigning” or digital signature verification process. Then, if the signature verification was successful, the unsigned code is

> compared to the computed code on the loaded program. The program is not run if either the signature verification fails, or if the code is incorrect.

Id., at 9:13-21.

409. The public-key digital signature technique disclosed in the Jablon Patent is a well-known technique from the computer security company RSA, which Jablon references in U.S. Pat. No. 4,405,289. *Id.* at 5:60-68. RSA digital signing relies on having a signor use a (secure) private key to encrypt a message or to encrypt a modification detection code. A party wishing to verify the authenticity of the signature can use a (widely available) public key to decrypt the modification detection code. If the public key is able to decrypt the modification detection code encrypted using sender's (trusted authority's) private key, the program can be considered verified because only the signing party (in Jablon, a trusted authority) could have signed the modification detection code. If the decrypted modification detection code matches the modification detection code of the program of interest, the program's integrity is verified and considered received as sent. *Id.*

410. The Jablon Patent discloses two ways in which the original modification detection codes and/or public keys can be written into the memory of the computer workstation. The first is through the use of a reconfiguration tool that operates in a special mode called "config mode" that is used to write a modification detection code and/or public key into the erasable non-volatile memory:

> But at least one reason for needing config mode, where the validation data can be changed, is to allow the system to be upgraded with a new boot record, as part of a general installation of new software.
>
> This "config" mode should only be used in these relatively uncommon circumstances, and the mode should only be available for authorized people. One way to control access to config mode is to create a special reconfiguration diskette, only to be possessed by the authorized people. The reconfiguration disk contains a tool which puts a new user-mode boot record on the hard disk, and which writes a new modification detection code into protectable non-volatile

memory.

Id., at 14:39-51 (emphasis added). This allows modification detection codes to be written into memory. It would have been readily apparent to a POSITA that this process would also write a public key to the memory in embodiments that use digital signatures, because Jablon anticipates that stored public keys may need to change:

> It is assumed that the authority's public-key will change much less frequently than the authorized software will change. . .

Id., at 19:45-47.

411. One further disclosure in the Jablon Patent is the use of a network server in order to distribute both software programs and associated modification detection codes, which it calls a "network bootstrap method." *Id.* at 18:40-66. In this embodiment, the PC workstation is diskless, and so lacks bulk persistent storage. *Id.*, at 18:40-41. Instead, the workstation requests access to a network server to receive bootstrap program files over the network. *Id.*, at 18:45-49. Program files retrieved over the network are validated using the same digitally-signed modification detection code and public-key digital signature algorithm described above. *Id.*, at 18:64-66.

412. For my analysis above I have relied on parts of the Jablon Patent whose disclosures are essentially the same as those in the Arbaugh Patent. The references of course have some differences but, under my current understanding of Ancora's allegations, those differences do not meaningfully impact my opinion. In particular, I note that the specification of the Arbaugh Patent asserts that the differences between the Arbaugh Patent and the Jablon Patent are limited to verification of the integrity of the BIOS itself and expansion ROMs, neither of which is implicated in Ancora's infringement contentions:

> Two patents, U.S. Pat. No. 5,379,342 to Arnold ("the Arnold patent") and U.S. Pat. No. 5,421,006 to Jablon ("the Jablon patent") also present secure boot models. Both of these patents are similar in that the BIOS verifies the boot block before control is transferred and the boot block verifies the OS kernel before

> control is transferred. The Jablon patent continues to provide Static integrity checks while the operating System is running, i.e validating the integrity of a program before execution. Another difference between the two patents is that the Arnold patent uses a Modification Detection Code, *e.g.* MD5, and Jablon uses public key cryptography. Both approaches, however, fail to verify the BIOS beyond the normal eight bit additive CRC, and both approaches also fail to verify expansion ROMs.

Arbaugh Patent, at 2:52-65.

XV. THE JABLON PATENT IS PRIOR ART

413. The Jablon Patent was filed on April 20, 1994 and issued on May 30, 1995, and claims priority to an earlier utility patent application, U.S. Pat. App. 10/880,050 filed on May 7, 1992. The ’941 Patent was filed on October 1, 1998, claiming priority to an Israeli patent

414. application filed on May 21, 1998. Regardless of the priority date claimed by Ancora, because the Jablon Patent issued more than one year before the earliest filing date of the ’941 Patent in the United States, qualifies as prior art to the ’941 Patent on the basis of pre-AIA 35 U.S.C. § 102(b).

415. Because the Jablon Patent issued on May 30, 1995, and the earliest filing date of the ’941 Patent in the United States is on October 1, 1998, the Jablon Patent qualifies as prior art under 35 U.S.C. § 102(b).

XVI. THE JABLON PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE ’941 PATENT

416. It is my opinion that the Jablon Patent anticipates claims 1-3, 6-14, and 16 of the ’941 Patent. For the reasons I discuss below, the Jablon Patent qualifies as prior art under 35 §§ 102 and 103. Each limitation of claims 1-3, 6-14, and 16 of the ’941 Patent is also disclosed by the Jablon Patent, either literally or inherently, or is obvious in view of the Jablon Patent.

A. Claim 1 is Anticipated by the Jablon Patent

1. Limitation 1[preamble] is Disclosed by the Jablon Patent

417. Limitation 1[preamble] of the '941 patent recites:

> 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:

418. I understand that the Court determined that the portion of the preamble reciting "a method of restricting software operation within a license" is non-limiting, and therefore does not need to be considered for purposes of invalidity. The Court determined that the portion of the preamble reciting "a computer including an erasable, non-volatile memory area of a BIOS of the computer, and a volatile memory area" is limiting.

419. Jablon discloses the computer of the preamble as claimed. Jablon discloses a computer as shown in Figure 1, which includes an "erasable, non-volatile memory area" 34 highlighted with yellow, a read-only memory area 15 which I have highlighted with blue, and a volatile memory area 13, highlighted in violet (right) (originally annotated by Dr. Nettles). Jablon Patent, at Fig. 1. The Jablon Patent explains that the BIOS of the computer is stored in ROM 15:

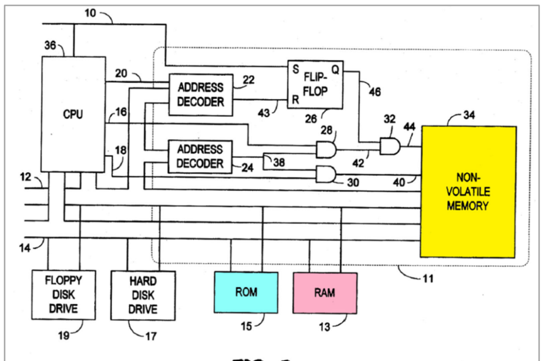


FIG. 1

> BIOS resides in read-only memory, and it is the first program to run when the CPU starts up.

Id., at 11:56-57. This BIOS then accesses and stores data in the non-volatile memory 34:

> The new verification process in the BIOS startup program is shown in FIG. 2, and it uses uses [sic] data stored in protectable non- volatile memory FIG. 1-34, as

> detailed in FIG. 3.

Id., at 11:67-12:2 (emphasis added). The protectable non-volatile memory may be written to when a write-protect latch is open, but cannot be written to when the write-protected latch is closed:

> A detailed description of the latch is found below, but the net effect of closing the latch is to prevent all subsequent programs from modifying the data shown in FIG. 3. . . .
>
> Thus, the non-volatile memory is initially writable after a system reset. At this point the latch is said to be open. . . .
>
> When the latch is closed, no software process is capable of modifying the contents of the data in 34, which is used to verify the integrity of the boot record.

Id., at 12:49-53, 14:4-6, and 14:23-25. A POSITA would have understood that a memory that is rewritable is necessarily erasable, since a rewrite operation in a non- volatile memory is often implemented as an erase operation followed by a write operation on the memory. Even though there may be some instances where the configuration of the device prevents the writing of new data to the non-volatile memory 34, it does not change the fact that the memory itself is erasable.

420. A POSITA would have also understood that RAM 13, as the memory in a "standard PC," refers to the volatile random-access memory typically used as working memory in a personal computer at the time:

> The standard PC is herein modified with additional hardware, in the form of a plug-in card shown in FIG. 1, and software changes made within the ROM BIOS program as shown in FIG. 2.

Id., at 11:14-18.

421. This conclusion still holds even under Ancora's argument regarding the scope of the "memory of the BIOS" in its POPR. There, Ancora argued that:

> Absent a showing that the BIOS module actually uses another portion of such physical memory space as part of its normal operations, only the BIOS module would be understood by a person of ordinary skill to be "memory of the BIOS."

Id., at 30. Regardless of whether that interpretation is correct, the Jablon Patent discloses that the non-volatile memory 34 is used to store data during the normal operation of the BIOS:

> But our focus here is on enhancements to the BIOS program, and on a new verification process that occurs between the time when BIOS loads the boot record program into memory, and the time when BIOS runs the boot record program. The new verification process in the BIOS startup program is shown in FIG. 2, and it uses uses [sic] data stored in protectable non-volatile memory FIG. 1-34, as detailed in FIG. 3.

Id., at 11:63-12:2. Because the non-volatile memory 34 is used by the BIOS in its normal operation, Jablon's non-volatile memory 34 discloses a "memory of the BIOS" under Ancora's interpretation of the term.

2. Limitation 1[a] is Disclosed by the Jablon Patent

422. Limitation 1[a] of the '941 patent recites:

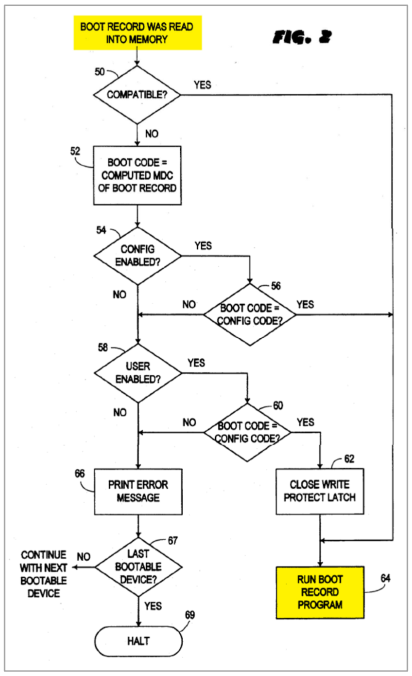
> selecting a program residing in the volatile memory

423. I understand that Ancora contends that the limitation of "selecting a program" is met by loading a program into RAM. Ancora Final Infringement Contentions Ex. B, at 19-20. Based on my review of those contentions, Ancora appears to contend that "selecting a program" is met by loading a program into RAM, even though the program is not present in RAM at the time of "select[ion]":

> For example, each Accused Android Product is configured by LGE such that, following its receipt of an OTA update from an Accused OTA Product, the Accused Android Product selects the OTA update and loads it into RAM.

Id. at 35.

424. The Jablon Patent discloses loading a program into a volatile memory, such as the RAM disclosed as the main memory of the PC in the Jablon Patent. The Jablon Patent illustrates as key steps in each of its embodiments the steps of loading a program into memory, verifying its modification detection code, and then (on a successful verification) executing the program loaded into memory:



Id. at Figure 2 (emphasis originally added by Dr. Nettles). In Figure 2 of the Jablon Patent, the flow chart begins by showing the boot record program being loaded into memory. The final step, after all verifications are completed, is execution of the boot record in memory. This is confirmed by the specification and claims of the Jablon Patent, that describe the same process. See, *e.g.*, Jablon Patent, claim 12 (disclosing verifying a second program in volatile memory and then executing it).

3. Limitation 1[b] is Disclosed or Rendered Obvious by the Jablon Patent

425. Limitation 1[b] of the '941 patent recites:

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

426. As I described above for the Arbaugh Patent, I understand that the Court has construed the following two terms relevant to this limitation:

> "using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS" to have its plain and ordinary meaning, and that the plain and ordinary meaning of the term "agent" is "a software program or routine."
>
> "set up a verification structure" to mean "establishing or certifying the existence of a pseudo-unique key and establishing at least one license-record location."[15]

427. I understand that Ancora contends that the limitation of a "license-record" is met by the presence of a hash value, cryptographic digest, or cryptographic hash used to verify the integrity of a program file or program file. Ancora Final Infringement Contentions Ex. B, at 28-50. Based on my review of those contentions, Ancora appears to contend that "setting up a verification structure" is met by confirming the existence of a pseudo-unique key and creating a license-record by storing a cryptographic hash that has been encoded with a private key:

> As the above reflects, such structure includes at least one license record, *i.e.*, data associated with the update package that allows the Accused Product to verify the update package, including payload information such as the application name or version of the program and/or a "hash" representation of such information that allows the program to be verified.

Id. at 37. This passage, as well as others in Ancora's infringement contentions, indicate that the presence of a hash representation of information used for verification suffices to meet this

[15] As mentioned for the Arbaugh Patent, above, the Court also determined that "Establishing at least one license-record location" may include the steps of "forming a license-record by at least partially encrypting the contents used to form a license-record with other predetermined data contents, using at least part of the pseudo-unique key; and storing the encrypted license-record," but that this clarification is not for presentation to the jury.

limitation.

428. I further understand from Ancora contends that storing this license-record information in a memory of a device that can be accessed from the BIOS suffices to meet the limitation that the verification structure be "in the erasable, non-volatile memory of the BIOS." Ancora's contentions identify various "cache" and "data" partitions that Ancora contends are "BIOS" memories, but there is no reference in any of the documents that the memory being used is BIOS memory. *Id.*, at 29 ("[The LGE device] downloads this package and saves it to the cache or data partition of its BIOS").

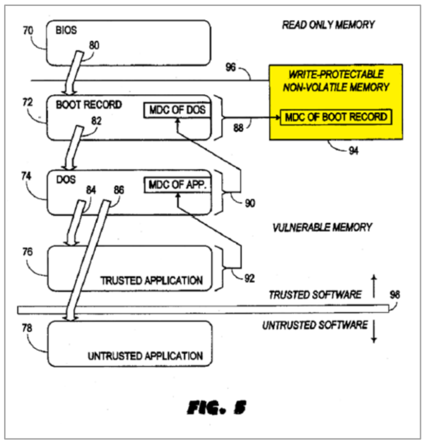
429. The Jablon Patent discloses using a public-key digital signature for verifying the integrity and authenticity of software. Individual trusted authorities can digitally sign different trusted software on the system. Jablon Patent, at 19:2-5. As explained by Jablon, when authorized software is first configured, a modification detection code is generated, and this modification detection code is then digitally signed using the private key of a trusted authority:

> The public-key digital signature variation is shown in FIG. 7. During configuration of an authorized program 152, a modification detection code is first computed 164, as before. But then a public- key digital signature is constructed for this modification detection code 166, using the private-key of a trusted authority 168, and this signature is stored associated with the authorized software 154, possibly appended to the end of the file.

Id., at 19:28-35. The authorized software is then verified by storing a public key of the trusted authority in the non-volatile memory of the computer system:

> The corresponding public-key of this authority 162 is then stored in the higher-level component 160, rather than storing the modification detection code of the authorized program.
> …
> In order to achieve maximal flexibility of reconfiguration, the public-key might be stored in the protected non-volatile memory FIG. 5-94, and the bootstrap ROM would perform a digital signature verification, as opposed to the plain modification detection code comparison currently shown in FIG. 2, steps 56 and 60.

Id., at 19:35-55. The non-volatile memory 94 is shown in Figure 5 of the Jablon Patent to be a write-protectable (*i.e.* erasable) non-volatile memory (right) (*Id.*, at Figure 5, 16;45-56 (emphasis originally provided by Dr. Nettles)).



430. The Jablon Patent discloses that this license record is set up by an "agent." As a preliminary matter, the Jablon Patent discloses the use of a software tool to write the modification detection code to the protectable non-volatile memory through the use of a special operating mode called "config mode" for installation of new software:

> But at least one reason for needing config mode, where the validation data can be changed, is to allow the system to be upgraded with a new boot record, as part of a general installation of new software.
>
> This "config" mode should only be used in these relatively uncommon circumstances, and the mode should only be available for authorized people. One way to control access to config mode is to create a special reconfiguration diskette, only to be possessed by the authorized people. The reconfiguration disk contains a tool which puts a new user-mode boot record on the hard disk, and which writes a new modification detection code into protectable non-volatile memory.

Id., at 14:39-51 (emphasis added). This process shows that what Ancora contends is a "license-record" is written to an erasable, non-volatile memory of the BIOS by an agent, in this case, the reconfiguration disk tool.[16] In the public-key embodiment disclosed in the Jablon Patent, a public key is written into the protectable non-volatile memory, and digitally signed modification

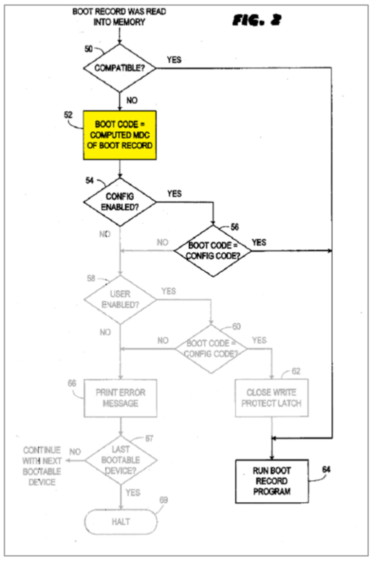
[16] This is similar to Ancora's contentions that the execution of a recovery system to install new boot software in the accused products can meet this limitation. See Ancora Infringement Contentions, Ex. B, at 59.

detection code are written into a memory in such a way that they can be "associated with the authorized software" that is to be executed. Under Ancora's asserted claim construction position, the location where this data is written is "memory of the BIOS" because it is memory used to store these components that are accessible by the BIOS. This limitation is explicitly disclosed by Jablon, but even if it were not, this limitation would have been rendered obvious to a POSITA in view of Jablon. The use of an "agent," which can be invoked as part of the BIOS or part of the operating system, would have been obvious to a POSITA at the time in view of Jablon, as well as other types of "agents" (including API calls) described in the DMI specification that were known to a POSITA at the time. *See*, Section IX.C, above.

431. The Jablon Patent also discloses an "agent" under at least Ancora's proposed claim scope in the form of the network bootstrap program which can be used in a diskless workstation to load the necessary configuration into the memory of the PC workstation in Jablon. *Id.* at 18:38-67. In this embodiment the PC workstation loads the software used to operate over a network connection:

> the security of diskless PC workstations on a network. In these machines the BIOS startup program is much the same, except that it initializes a network device, and loads the boot record and all subsequent programs across a network, rather than from a local disk.

432. *Id.*, at 18:40-45. This modified version of the secure boot process in the Jablon Patent is configured to remove steps 58, 60, and 62, with the step of verification of the modification detection code of the boot record remaining, as shown in the below annotated Figure 2 (right) (*Id.*, at Figure 2 (originally annotated by Dr. Nettles to remove execution paths unavailable in network boot)).



433. The network bootstrap embodiment is otherwise the same as the Fig. 2 embodiment discussed earlier. Jablon Patent, 18:50-56. The network bootstrap embodiment therefore discloses the establishment of the modification detection code in the erasable, non- volatile memory. The Jablon Patent discloses that the network boot method uses the "trusted path login" model, which is that:

> When the system starts, the BIOS startup program loads, verifies, and runs the boot record program as described in the previous embodiment. This is a new boot record enhanced with code that will verify DOS before running it.

Jablon Patent, 16:57-61. In the network bootstrap embodiment, then, the boot record is loaded over a network, and verified against the modification detection code for the boot record.

434. The Jablon patent discloses that the modification detection code was placed in the non-volatile memory when the system was initialized:

> The first step is to compute the modification detection code for the boot record to be used in normal secure operation. The computed modification detection code value is written into userCode 112 in the protectable non-volatile memory.

Id., at 15:35-40.

435. The Jablon Patent discloses that one type of program that may be verified using

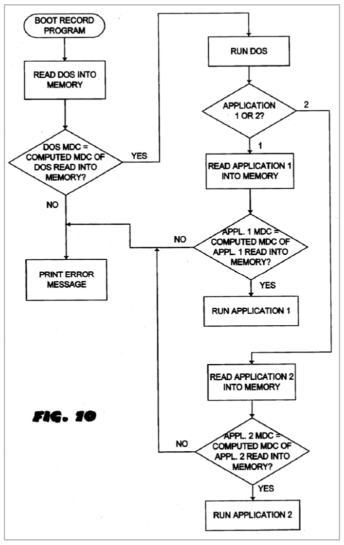
its method before the program is run is a "system initialization program", *i.e.*, BIOS program. Jablon Patent, 8:39-42. Moreover, the Jablon Patent discloses that the "boot record" software component is subject to verification by BIOS. Jablon Patent, 16:57-61. The BIOS, when verifying the boot record software component, accesses both the storage area for the boot record software component, and the write-protectable non-volatile memory where the MDC of the boot record is stored. Jablon Patent, Fig. 5. Under Ancora's claim construction position, "memory of the BIOS" is memory accessible by the BIOS as part of its normal operations. For that reason, both the storage area of the boot record component and the write-protectable non-volatile memory that stores the MDC of the boot record are "memory of the BIOS."

436. In the public-key digital signature embodiment of the Jablon Patent, a public key is stored in the protectable non-volatile memory instead of storing a modification detection code therein, *id.* at 19:50-55. And a digital signature is stored with the software component, for example, the boot record. *Id.* at 19:31-35, Fig. 5. A POSITA would have understood that the upgrade process, when applied using the public-key digital signature embodiment, would write the public key into the protectable non-volatile memory. The digitally-signed modification detection codes for each program to be executed would similarly be downloaded and written, for example, into the storage area of the updated boot record software component. *Id.* at 19:31-35, Fig. 5. The digital signature would be "associated with the authorized software", *e.g.*, the boot record, that is to be executed. *Id.* Thus, under Ancora's claim construction position, both the digital signature and the public key are stored in "non- volatile memory of the BIOS" (because both the storage area of the boot record component and the write-protectable non-volatile memory that stores the MDC of the boot record are "memory of the BIOS", as discussed immediately above).

437. The Jablon Patent also discloses that program code in the operating system can invoke its verification procedure in order to confirm that application codes executed by the operating system have not been tampered with:

> FIG. 10 illustrates the flow of the process after the boot record program is executed for this alternative. Note that DOS loads, verifies and executes both applications.

Jablon Patent, 17:55-58. Figure 10 shows an execution process where the operating system invokes an application to execute, and during that execution, the modification detection code of the application is computed and verified against a stored modification detection code (right) (Jablon Patent, Fig. 10). This process utilizes the public key that is stored in the protectable non- volatile memory 34 in the public key digital signature embodiment of the Jablon Patent, which I understand Ancora contends meets this limitation. See Arbaugh Patent claim 1[b], supra. This embodiment also uses the digitally-signed modification detection code in order to verify the application program, as described earlier with respect to this limitation.


BOOT RECORD PROGRAM
READ DOS INTO MEMORY
DOS MDC = COMPUTED MDC OF DOS READ INTO MEMORY?
YES
NO
PRINT ERROR MESSAGE
RUN DOS
APPLICATION 1 OR 2?
1
2
READ APPLICATION 1 INTO MEMORY
APPL. 1 MDC = COMPUTED MDC OF APPL. 1 READ INTO MEMORY?
NO
YES
RUN APPLICATION 1
READ APPLICATION 2 INTO MEMORY
APPL. 2 MDC = COMPUTED MDC OF APPL. 2 READ INTO MEMORY?
NO
YES
RUN APPLICATION 2
FIG. 10

438. To the extent that this claim term has the scope that Ancora urges, this limitation is disclosed by either client software or server software in the Jablon Patent, The Jablon Patent discloses a bootstrap server software that provides digitally signed cryptographic hash of a program, which digitally signed cryptographic hash is transferred to the client in Jablon and stored in the non-volatile memory of the BIOS. As described in the Jablon Patent, an authority exists that provides digitally-signed modification verification codes corresponding to programs, and these can be transferred to a client by a bootstrap server:

> In an embodiment that uses public-key encryption, it is necessary that the signature generation process that occurs during configuration be done on a secure system.

Jablon Patent, 19:56-58.

> Another variation of this embodiment enhances the security of diskless PC workstations on a network. In these machines the BIOS startup program is much the same, except that it initializes a network device, and loads the boot record and all subsequent programs across a network, rather than from a local disk.

Jablon Patent, 18:40-45. Note that Jablon does not require, nor does Ancora contend, that the "agent" is necessarily software operating on only one computer, so under the broad claim scope that Ancora has asserted the use of one server to digitally sign and another server to transfer the digitally signed modification detection code and software can still comprise a single "agent," as I understand Ancora's assertions. To the extent that Ancora is asserting a claim scope that encompasses software operating on a server as an "agent" meeting this limitation, therefore, that limitation is disclosed by the Jablon Patent disclosure of one or more servers using software to generate a private key, digitally signing a modification detection code of a program, and transferring the digital signature and the program to a client.

439. This conclusion still holds in view of Ancora's argument regarding the scope of the "memory of the BIOS" in its POPR. There, Ancora argued that:

> Absent a showing that the BIOS module actually uses another portion of such physical memory space as part of its normal operations, only the BIOS module would be understood by a person of ordinary skill to be "memory of the BIOS."

Id., at 30. Regardless of whether that interpretation is correct, the Jablon Patent discloses that the non-volatile memory 34 is used to store data during the normal operation of the BIOS:

> But our focus here is on enhancements to the BIOS program, and on a new verification process that occurs between the time when BIOS loads the boot record program into memory, and the time when BIOS runs the boot record program. The new verification process in the BIOS startup program is shown in FIG. 2, and it uses uses [sic] data stored in protectable non-volatile memory FIG. 1-34, as detailed in FIG. 3.

Id., at 11:63-12:2. Because the non-volatile memory 34 is used by the BIOS in its normal operation, the non-volatile memory 34 is a "memory of the BIOS" under Ancora's interpretation of the term. Similarly, the memory used to store the modification detection codes and/or digitally-signed modification detection codes for each program to be executed are also stored in a memory by this verification process, which is managed by the BIOS. Accordingly, each memory so-used is a "memory of the BIOS" under Ancora's asserted claim scope.

440. This limitation is explicitly disclosed by the Jablon Patent, but even if it were not, this limitation would have been rendered obvious to a POSITA in view of the Jablon Patent. It would be obvious to use a software program as an "agent" both at the BIOS and operating system level. The Jablon Patent discloses the execution of a boot record verification process, shown in Figure 2, and an operating system verification process, shown in Figure 10, that also verifies the modification detection codes of programs that are executed on the platform. It would have been obvious to a POSITA that the program code can be executed at the level of the BIOS during configuration of the boot record, *e.g.*, Jablon Patent at 12:27-49, at the level of the operating system during execution of an installation tool on a reconfiguration diskette, *e.g.*, Jablon Patent at 14:52-68, or as a process within the operating system to confirm that an application is verified before execution. *E.g.*, Jablon Patent, 17:55-59. It would accordingly have been obvious to a POSITA to apply the technique disclosed for the verification of the boot record and operating system to verification of programs invoked by the operating system as an operating-system level verification with digital signatures for the applications to be verified stored in the non-volatile memory area 34. This modification could have been achieved by a POSITA without undue experimentation because the Jablon Patent already provides a road map for implementing such a change, and applying the programming techniques used by Jablon to

other programs within the software hierarchy would be nothing more than applying known techniques to similar problems in the same field which would yield the expected result.

441. It would also be obvious to store public-key digital signatures for modification detection codes in the protectable non-volatile memory. The Jablon Patent discloses in its public key digital signature embodiment that modification detection codes are calculated for each program, and are stored in a way that is associated with the program:

> During configuration of an authorized program 152, a modification detection code is first computed 164, as before. But then a public- key digital signature is constructed for this modification detection code 166, using the private-key of a trusted authority 168, and this signature is stored associated with the authorized software 154, possibly appended to the end of the file.

Jablon Patent, 19:29-35. A POSITA would have understood that the instruction in Jablon that the signature is stored "associated with the authorized software" leaves open the possibility of storing the signature in different locations depending on the particular needs of the system configuration. Moreover, the statement that the signature could possibly, but not necessarily, be appended to the program file would have indicated to a POSITA that the signature could be stored somewhere other than appended to the end of the program file of an authorized program 152. In particular, the improvement described in column 19 uses a digitally-signed modification detection code in order to avoid having a change to one modification detection code in the system result in cascading re-computation of each modification detection code that was calculate over that modification detection code:

> Recalling the discussion of how the hierarchy of modification detection codes is configured, the avalanche of code changes that are necessary whenever a trusted program changes may sometimes be a problem.

Jablon Patent, 19:6-9. As a POSITA would have realized, because Jablon acknowledges it, using a chain of modification detection codes where each program includes the modification detection codes of each program it invokes means that if a high-level program changes, its modification detection code changes, as does the program that invokes that program, and so on. Therefore the modification detection code for the boot record must be recalculated every time any program changes. Using the public key digital signature system avoids this situation even if all the signatures are stored in the protectable non-volatile memory, because each modification detection code is not calculated over the modification detection codes for other programs. With this problem solved, a POSITA would have recognized that storing the digitally signed modification detection code in the protectable non- volatile memory area would be an obvious design choice. The advantage of storing the digital signature in the protectable non-volatile memory is that it is more secure, as taught by Jablon, precisely because it is more difficult to modify and more resistant to attack. Jablon Patent, 10:32-34. This modification could have been achieved by a POSITA without undue experimentation because the Jablon Patent already provides a road map for implementing such a change, and applying the programming techniques used by Jablon to other programs within the software hierarchy would be nothing more than applying known techniques to similar problems in the same field which would yield the expected result.

4. __Limitation 1[c] is Disclosed by the Jablon Patent__

442. Limitation 1[c] of the '941 patent recites:

> verifying the program using at least the verification structure from the erasable non-volatile memory of the BIOS

443. As discussed earlier with respect to the Arbaugh Patent, I understand that the Court construed the term "verifying the program using at least the verification structure" to mean "confirming whether a program is licensed using at least the verification structure." In the context of the present litigation, I understand that Ancora contends that a program being "licensed" means that the program is authorized to be executed.

444. The Jablon Patent discloses a verification process that uses the public key stored in the protected non-volatile memory:

> During verification of the authorized program 152 the higher-level component 150 will compute the modification detection code for the program, and then, using the stored public-key 160, it will verify that the digital signature as stored with the authorized software 154 is correct.

Id., at 39-44. This uses "the verification structure" because it uses the license record, *e.g.*, it uses the public key as well as the digital signature, stored in the memory of the computer system in the Jablon Patent.

445. The verification process in the Jablon Patent performs the task of "verifying" as used in the '941 Patent because the process confirms whether a program is licensed, *e.g.* authorized to be executed, based on the result of the verification:

> The verification process occurs in two steps: first the signed modification detection code is retrieved during the 'unsigning' or digital signature verification process. Then, if the signature verification was successful, the unsigned code is compared to the computed code on the loaded program. The program is not run if either the signature verification fails, or if the code is incorrect.

Id., at 9:14-21. Accordingly, if a verification does not complete successfully, by confirming that the signed modification detection code matches the calculated modification detection code, the code is not authorized to execute. That is, the code is not "licensed."

5. __Limitation 1[d] is Disclosed by the Jablon Patent__

446. Limitation 1[d] of the '941 patent recites:

> acting on the program according to the verification.

447. As I discussed earlier with respect to the Arbaugh Patent, I understand that the Court construed the term "acting on the program according to the verification" to have its plain and ordinary meaning, and that the plain and ordinary meaning includes that the step of "acting on the program" may include but is not limited to "restricting the program's operation with

predetermined limitations, informing the user on the unlicensed status, halting the operation of the program under question, and asking for additional user interactions."

448. The Jablon Patent discloses "acting on the program according to the verification" in the form of preventing the execution of programs whose integrity cannot be validated, *i.e.*, verified. As I discussed for the immediately preceding limitation, the Jablon Patent discloses that a program will not be permitted to execute if it fails verification. *Id.*, at 9:14-21 ("The program is not run if either the signature verification fails, or if the code is incorrect."). The Jablon patent also discloses informing the user of the unlicensed status in the form of displaying an error message in the event of a verification failure:

> If the boot record has been corrupted, step 52 of the initialization process in FIG. 2 will compute a different modification detection code value, recalling that it is practically impossible for two different programs to compute codes to the same value. The remainder of the initialization process will be the same as for a correct user boot record, up until test 60, which will fail. At this point an error message will be printed 66.

Id., at 13:3-10. These two situations described in the Jablon Patent—on the one hand, providing an error message, and on the other hand, terminating execution—match the behaviors encompassed in the Court's construction of the term "acting on the program according to the verification," and this limitation is disclosed by the Jablon Patent.

B. Claim 2 is Anticipated by the Jablon Patent

449. Claim 2 of the '941 patent recites:

> A method according to claim 1, further comprising the steps of: establishing a license authentication bureau.

450. The Jablon Patent renders claim 2 invalid because it discloses all of the limitations discussed above for claim 1, from which claim 2 depends, and also discloses the step of "establishing a license authentication bureau" in dependent claim 2. As I discussed earlier with respect to this limitation of the '941 Patent in the context of the Arbaugh Patent, the

defining characteristics of a "license authentication bureau" are that it does one, or the other, or both of

> establishing the license record in the second non-volatile memory; and
>
> verifying if the key and license record in the non-volatile memory(s) is compatible with the license record information as extracted from the application under question.

Id., at 3:34-41. The function of verifying if the key and license record are compatible with the license record information extracted from the application under question are disclosed by what the Jablon Patent calls the "higher level component 150," shown in Figure 7 (right) (*Id.*, at Fig. 7). Here, and as shown in Figure 7, each of program B (152) and program C (no number assigned) includes a modification detection code (*e.g.*, MDC of B 154 for program B) that is signed by the trusted authority, referred to as "X" in Figure 7. During verification, higher-level component 150 will use the public key of trusted authority X 160 to verify the signature of the MDC, and then compare the signed-and-stored MDC with the derived MDC:

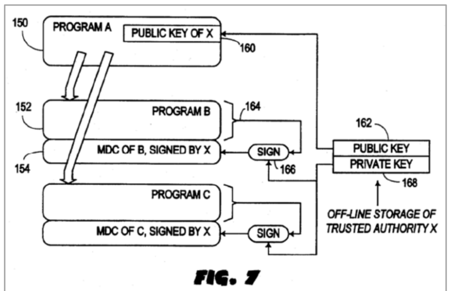


FIG. 7

> modification detection code of the authorized program. During verification of the authorized program 152 the higher-level component 150 will compute the modification detection code for the program, and then, using the stored public-key 160, it will verify that the digital signature as stored with the authorized software 154 is correct.

Id., at 19:38-44. The Jablon Patent therefore disclosed "establishing" a license authentication bureau, *e.g.* the "higher-level component 150," because it discloses that such a program is necessary and must be included to verify the license record of programs B and C. *Id.*

451. In addition, the Jablon Patent discloses the alternative establishment of a license authentication bureau that establishes the license record in the non-volatile memory. The license

record is the signed MDC present for each program, *e.g.* program B and C shown in Figure 7 of the Jablon Patent. The Jablon Patent further discloses that these license records can be generated on a second computer system, and are then placed in the non-volatile memory of the first computer system:

> In an embodiment that uses public-key encryption, it is necessary that the signature generation process that occurs during configuration be done on a secure system. For high-security installations, the signing process might occur off-line, on a completely separate machine.

452. *Id.*, at 19:56-60. This discloses establishing a license authentication bureau because in order to sign the MDC of programs to be configured on the protected computer system, there must be software that performs the "signing process," *e.g.*, generating the license record, and which ensures that the license record is present in the non-volatile memory of the protected computer system:

> In order to achieve maximal flexibility of reconfiguration, the public-key might be stored in the protected non-volatile memory FIG. 5-94, and the bootstrap ROM would perform a digital signature verification, as opposed to the plain modification detection code comparison currently shown in FIG. 2, steps 56 and 60.

453. *Id.*, at 19:50-55.

C. Claim 3 is Anticipated or Rendered Obvious by the Jablon Patent

1. ***<u>Limitation 3[a] is Disclosed by the Jablon Patent</u>***

454. Limitation 3[a] of the '941 patent recites:

> A method according to claim 2, wherein setting up a verification structure further comprising the steps of establishing, between the computer and the bureau, a two-way data-communications linkage

455. The Jablon Patent discloses one embodiment where the verification process is performed over a network connection between a PC workstation and a network server. See Jablon Patent, at 18:40-66. This embodiment is explicitly described as being compatible with the

embodiment that uses the public-key digital signature verification that I rely on for other limitations in the '941 Patent. In this embodiment, the PC workstation communications with a network server.

456. In this embodiment, the Jablon Patent describes that the PC workstation receives a new user-boot mode record, which includes the modification detection code for programs to be executed. This is described as being available in "config mode," which is ordinarily accomplished by way of using a "reconfiguration disk":

> The reconfiguration disk contains a tool which puts a new user- mode boot record on the hard disk, and which writes a new modification detection code into protectable non-volatile memory.

Id., at 14:48-51. In the Jablon Patent's network model embodiment, however, there is no diskette drive, so new configurations need to be obtained over the network. *Id.*, at 18:56-59 ("Since there is no local diskette drive, and hence no reconfiguration diskette, the previously described trusted-path login mechanism should be used to allow software reconfiguration.").

457. A person of ordinary skill in the art would recognize that this disclosure in Jablon describes the computer (*i.e.* the PC workstation in Jablon) establishing a two-way data-communications linkage (*i.e.* a network connection in Jablon) with a license bureau (*i.e.* software running on a server in Jablon). The server in Jablon provides information including a new modification detection code to the PC workstation, which can then be used for subsequent verification by the PC workstation.

2. __Limitation 3[b] is Disclosed or Rendered Obvious by the Jablon Patent__

458. Limitation 3[b] of the '941 patent recites:

> 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

459. The Jablon Patent discloses this limitation in the same passage describing a

network bootstrap method that I referenced in the preceding limitation. Specifically, the Jablon Patent describes loading a boot record and programs over a network connection, rather than locally from a disk in the PC workstation:

> In these machines the BIOS startup program is much the same, except that it initializes a network device, and loads the boot record and all subsequent programs across a network, rather than from a local disk.

Id. at 18:40-45.

460. A POSITA would have understood that in order for a PC workstation to load a boot record from a server over a network, the PC workstation must provide an identification. At the lowest level it is necessary to provide a network identification in the data packets transmitted to the server, so that the server can authenticate the request to load data remotely. This network identification is also required so that the server can respond to the request to load data by transmitting data packets back to the PC workstation. Without an identification of the computer, such as an IP address, the server would have no way of transmitting requested data back to the PC workstation.

461. A POSITA would have recognized that an IP address, or other address used to uniquely identify a PC workstation on a network, would serve as an "identification of a computer." No party requested construction of the term "identification of a computer" so I understand the term should be given its plain and ordinary meaning within the art. As such an address that uniquely identifies the requesting computer for the server, such that the server can determine which computer is transmitting the request, would satisfy this limitation. The '941 Patent contemplates the use of information such as an IP address as an "identification" when it discusses the license bureau:

> The actual key that serves for identifying the computer may be composed of the pseudo-unique key exclusively, or, if desired, in combination with information, *e.g.* information related to the registration of the user such as *e.g.* place, telephone

> number, user name, license number, etc.

’941 Patent, at 4:6-10. An IP address is similar to a telephone number in that it allows the authority that is preparing the license to have some way to verify whether the entity requesting the license is authorized to do so. A POSITA would have recognized that a network address satisfies this limitation.

462. Alternatively, a POSITA would have considered it obvious that the system described in Jablon would need to transfer an identification of the computer to the server in order to implement the network bootstrap method described in column 18 of Jablon. It was well-known among skilled artisans that when using a network bootstrap technique that some piece of identifying information should be sent to the server servicing the bootstrap request so that configurations can be tailored to specific computer machines. For example, at least ten years before the Jablon Patent, the bootstrap protocol (BOOTP) was already being used, which transferred a unique hardware identifier of a computer system when requesting boot files. *See*, *e.g.*, IETF RFC 951: Bootstrap protocol (BOOTP, September 1985) at 5-8. This is because it was customary in the art at the time for diskless workstations to have customized boot software, for example to accommodate different hardware configurations on individual workstations.

463. The Jablon Patent also discloses transferring the license-record's contents from the selected program. The Jablon Patent discloses using a network bootstrap method, which a POSITA would have understood would include well-known bootstrap techniques such as BOOTP. According to the ’941 Patent specification, the “license-record’s contents” can include:

> The volatile memory accommodates a license program (16) having license record fields (13-15) appended thereto. By way of example said fields stand for application names (*e.g.* Lotus 123), Vendor name (Lotus inc.), and number of licensed copies (1 for stand alone usage, >1 for number of licensed users for a network application).

’941 Patent, at 5:27-34; see *id.* at 5:46-49 (“This transaction includes the key (8), the encrypted license records (10–12), contents from the license program used in forming a license record (*e.g.* fields 13–15), and other items of information as desired."). A POSITA would have understood that the filename used by well-known bootstrap protocols like BOOTP to request boot programs are similar to the kinds of fields identified in the '941 Patent as being transferred to the server.

3. __Limitation 3[c] is Disclosed by the Jablon Patent__

464. Limitation 3[c] of the '941 patent recites:

> 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

465. I understand that Ancora contends that this limitation can be met by having a cryptographic signature applied to a digest of a computer program. Ancora Final Infringement Contentions Ex. B, at 85-93. Based on my review of those contentions, Ancora appears to contend that "forming an encrypted license record at the bureau by encrypting parts of the request-for-license" is met by using information from the program whose integrity is to be verified. Ancora references forming an encrypted license

> by encrypting parts of the request-for-license using part of the identification of the Accused Android Product and the license- record's contents from the selected program, including "build" information (https://developer.android.com/reference/android/os/Build), as an encryption key.

Id. at 85. I understand from this that in order to meet this limitation, according to Ancora, what must be shown is that an integrity verification code is derived from the program whose integrity is to be verified. I also understand that Ancora contends that "using part of the identification as an encryption key" can be met by using a private key of a trusted authority to sign the relevant integrity certification, and using a public key to verify the signature. *Id.* at 86-93.

466. Based on Ancora's assertions regarding the scope of this limitation, the Jablon Patent discloses this limitation. The Jablon Patent discloses that when programs are retrieved from a network host, they include a modification detection code that has been digitally signed

using the private key of a trusted authority. Jablon Patent, 19:28-38.

467. A POSITA would have understood that, consistent with Ancora's infringement allegations, the modification detection code is "form[ed] . . . at the license bureau" because the Jablon Patent discloses that the digitally signed modification detection code should be generated at the trusted authority. Jablon Patent, 19:28-49.

468. A POSITA would have understood that, consistent with Ancora's infringement allegations, the modification detection code "encrypt[s] part of the request-for-license" because the request-for-license identifies a program that is to be downloaded. A POSITA would have understood that a cryptographic hash of the contents of a file serves to "identif[y]" the relevant binary, because it is a unique number that is associated with a given program.

469. A POSITA would have understood that, consistent with Ancora's infringement allegations, the modification detection code digitally signed with the private key of the trusted authority "us[es] part of the identification as an encryption key." Ancora contends that signing with a private key of a trusted authority meets this limitation, notwithstanding that the private key of a trusted authority is not part of an "identification of a computer," as this claim requires. This contention appears to be based on the assertion that a private/public key pair can be considered two parts of a single encryption key, and that the use of a private/public key pair to sign a given program means that the encryption keys identify that program. Ancora Final Infringement Contentions, Ex. B, at 85-93. Assuming Ancora's claim construction position that the identification of a program is deemed to identify a private/public key pair on a trusted authority, then the Jablon Patent discloses this limitation because it discloses using an appropriate private key at the trusted authority to digitally sign the modification detection code for each program that is requested:

> During configuration of an authorized program 152, a modification detection code is first computed 164, as before. But then a public- key digital signature is constructed for this modification detection code 166, using the private-key of a trusted authority 168, and this signature is stored associated with the authorized software 154, possibly appended to the end of the file.

Jablon Patent, at 19:29-35. The Jablon Patent contemplates that this signing process can occur on a "completely separate machine" from the computer implementing the secure boot process:

> In an embodiment that uses public-key encryption, it is necessary that the signature generation process that occurs during configuration be done on a secure system. For high-security installations, the signing process might occur off-line, on a completely separate machine.

Id., at 19:56-60.

4. ***<u>Limitation 3[d] is Disclosed by the Jablon Patent</u>***

470. Limitation 3[d] of the '941 patent recites:

> transferring, from the bureau to the computer, the encrypted license record

471. The Jablon Patent discloses transferring the encrypted license record to the computer. In the network bootstrap embodiment files are loaded remotely because there is no local disk. Jablon Patent, 18:40-45. Otherwise Jablon explains that the embodiment is the same as the standard PC workstation embodiment. *Id.*, at 18:50-56. A POSITA would have therefore understood that in the network bootstrap embodiment, with the public key digital signature embodiment applied, then the encrypted license record is downloaded as part of the software program over the network bootstrap protocol. This is because in the public-key model, the digitally-signed modification detection code may be appended to the program file itself. *Id.*, at 19:31-35. When a program is downloaded over the network bootstrap protocol, then, the program will already contain the license record. The license record is encrypted because, as noted, the modification detection code is digitally signed, that is, encrypted using the private key of the trusted authority. *Id.*

5. Limitation 3[e] is Disclosed by the Jablon Patent

472. Limitation 3[e] of the '941 patent recites:

> storing the encrypted license record in the erasable non-volatile memory area of the BIOS

473. I understand that Ancora contends that this limitation can be met by downloading the accused license record into a "cache" or "data" partition of a device. Ancora Final Infringement Contentions Ex. B, at 85-93. Based on my review of those contentions, Ancora appears to assert, in a conclusory manner, that the "cache" or "data" partition of the memory of accused LGE devices are part of the "BIOS." While Ancora does not explicitly state why it contends that the "cache" or "data" portions of the memory are necessarily part of the "BIOS," it may be based on the Court's construction of BIOS, which is:

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

Dkt. 69, Final Claim Constructions of the Court, at 2. Ancora's interpretation appears to be that any memory constitutes part of the BIOS if it contains code that is "run automatically when a computer is turned on" if it has some role in "start[ing] the operating system" or "support[ing] the transfer of data among hardware devices," even if the memory is not part of what is typically considered a BIOS by a person having ordinary skill in the art.[17]

474. Based on Ancora's assertions regarding the scope of this limitation, the Jablon Patent discloses this limitation. The Jablon Patent discloses that when programs are retrieved

[17] There is an apparent inconsistency between the argument that Ancora makes in its infringement contentions, that any memory storing data received that is used by the BIOS can constitute a "memory for the BIOS," and its statements in its POPR regarding the limited circumstances in which a memory can be considered a "memory for the BIOS," see Section VIII.E, above. Ancora's POPR arguments are addressed later in this section.

from a network host, they are placed in memory that contains code executed when the computer is turned on, including to help start the operating system, as described below.

475. The Jablon patent discloses in its network bootstrap embodiment that the boot record and all subsequent code is loaded across a network rather than from local storage:

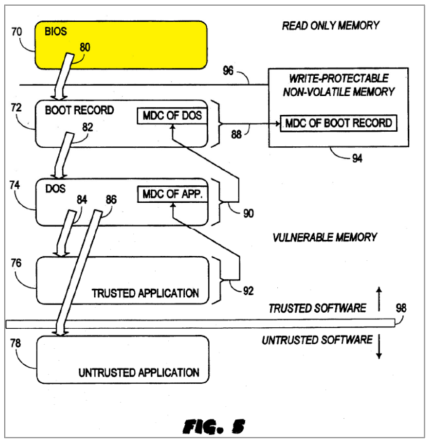
> In these machines the BIOS startup program is much the same, except that it initializes a network device, and loads the boot record and all subsequent programs across a network, rather than from a local disk.

Jablon Patent, 18:41-45. When the boot record (and subsequent programs) are loaded across a network, they are necessarily stored in a BIOS according to the Court's claim construction as applied by Ancora in its infringement contentions, because the boot record (and subsequent programs) are executed when the computer is turned on and help start the operating system, as explained earlier in the Jablon Patent:

> BIOS resides in read-only memory, and it is the first program to run when the CPU starts up. It performs some initial system checks, and then it loads the boot record program from disk into main memory, and runs the boot record program 80. The boot record program later loads and runs DOS, which in turn loads and runs applications.

Jablon Patent, 11:57-63. The boot record program would be considered part of the BIOS in this interpretation, and therefore (by extension) any memory used to store the boot record would constitute the "BIOS," under Ancora's infringement contentions and interpretation of claims.

476. A POSITA would have understood that the remotely- loaded boot record in the network bootstrap model with public key digital signatures, as described at 18:63-65 of the Jablon Patent, would include a modification detection code signed with the private key of the trusted authority. *Id.*, at 19:29-35. The boot record is an "authorized program," and a POSITA would have understood from Figure 5 of the Jablon Patent that the BIOS 70 is invoked to execute the boot record (right) (Jablon



patent, Fig. 5 (originally annotated by Dr. Nettles)). Figure 5, however, does not use the public key digital signature system. As described in the public key digital signature system, any instance of "trusted software" -- for example boot record 72, shown in Figure 5 -- can have its modification detection code appended to the program instead of being stored in the write-protectable non-volatile memory 94:

> In this improved method, we define a subset of the trusted software, called the "authorized software", which, when changed, will not propagate modification detection code changes through the rest of the trusted software, yet it will still be verified before it is run.

Id., at 19:23-27. The public key used for signature verification is instead stored in the write-protectable non-volatile memory 94:

> In order to achieve maximal flexibility of reconfiguration, the public-key might be stored in the protected non-volatile memory FIG. 5-94, and the bootstrap ROM would perform a digital signature verification, as opposed to the plain modification detection code comparison currently shown in FIG. 2, steps 56 and 60.

Id., at 19:50-55. Therefore, when the boot record is remotely loaded in the network bootstrap model, the boot record will include the modification detection code, as shown in Figure 7 and described at 19:1-60. This, according to Ancora, meets the requirement of being a "license-record," and the memory location into which the boot record is stored is an erasable non-volatile memory area of the BIOS because the memory area where the boot record modification detection code is stored is in the non-volatile memory area that stores certificates

477. This conclusion still holds even under Ancora's argument regarding the scope of the "memory of the BIOS" in its POPR. There, Ancora argued that:

> Absent a showing that the BIOS module actually uses another portion of such physical memory space as part of its normal operations, only the BIOS module would be understood by a person of ordinary skill to be "memory of the BIOS."

Id., at 30. Regardless of whether that interpretation is correct, the Jablon Patent discloses that the non-volatile memory 34 is used to store data during the normal operation of the BIOS:

> But our focus here is on enhancements to the BIOS program, and on a new verification process that occurs between the time when BIOS loads the boot record program into memory, and the time when BIOS runs the boot record program. The new verification process in the BIOS startup program is shown in FIG. 2, and it uses uses [sic] data stored in protectable non-volatile memory FIG. 1-34, as detailed in FIG. 3.

Id., at 11:63-12:2. Because the non-volatile memory 34 is used by the BIOS in its normal operation, the non-volatile memory 34 is a "memory of the BIOS" under Ancora's interpretation of the term.

D. Claim 6 is Anticipated by the Jablon Patent

478. Claim 6 of the '941 Patent recites:

> 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.

479. The Jablon Patent renders claim 6 invalid because it discloses all of the limitations discussed above for claim 1, from which claim 6 depends, and also discloses the step of "establishing a license authentication bureau" in dependent claim 6.

480. As I explained earlier in at least limitation 1[a], the Jablon Patent discloses loading a licensed-software-program into the volatile memory of the computer. A POSITA would have readily understood that the programs in the Jablon Patent are loaded into volatile memory in order to be executed, as was conventional in the art at the time.

481. Ancora in its infringement contentions has effectively alleged that a hash of a software component is part of the content of that software component. In particular, Ancora alleged that the hash carried in the VB Meta data structure is content of the software component. See Ancora Final Infringement Contentions Ex. B, at 100-104.

482. Based on such a broad interpretation of "said licensed-software-program includes contents used to form the license record" as recited in claim 6 (which interpretation I do not agree with), a POSITA would have undersood that the hash or MDC of a software component is content of the software component:

> During configuration of an authorized program 152, a modification detection code is first computed 164, as before. But then a public- key digital signature is constructed for this modification detection code 166, using the private-key of a trusted authority 168, and this signature is stored associated with the authorized software 154, possibly appended to the end of the file.

Jablon Patent, 19:29-35 (emphasis added). This passage of the Jablon Patent explains that a signature, which is a digitally signed cryptographic hash, is stored associated with, and possibly appended to, the program to which it is associated. The Jablon Patent is explicit that its signature is applied to the "modification detection code" of the authorized program 152 calculated in step 164, and that a modification detection code is a cryptographic hash:

> Modification detection codes are also known by other names in the literature, including: "cryptographic checksum", "cryptographic hash" , "secure hash algorithm", and "message digest".

Jablon Patent, 5:43-47. For that reason, if Ancora's asserted claim scope for purposes of infringement is assumed to be correct, then the Jablon Patent discloses that "said licensed-software-program includes contents used to form the license-record."

E. Claim 7 is Anticipated by the Jablon Patent

1. __Limitation 7[a] is Disclosed by the Jablon Patent__

483. Limitation 7[a] of the '941 patent recites:

> A method according to claim 6 wherein using an agent to set up the verification structure includes the steps of: establishing or certifying the existence of a pseudo-unique key in a first non- volatile memory area of the computer.

484. The agent in the Jablon Patent establishes or certifies the existence of a pseudo-unique key in a first non-volatile memory area.[18] The Jablon Patent discloses establishing or certifying the existence of a pseudo-unique key in the non-volatile memory of the BIOS, because that is how the Court construed the term "set up a verification structure."[19]

485. The Jablon Patent discloses establishing the presence of the public key for the trusted authority that signs the modification detection code in the higher-level program 150:

> During configuration of an authorized program 152, a modification detection code is first computed 164, as before. But then a public- key digital signature is constructed for this modification detection code 166, using the private-key of a trusted authority 168, and this signature is stored associated with the authorized software 154, possibly appended to the end of the file. The corresponding public-key of this authority 162 is then stored in the higher-level component 160, rather than storing the modification detection code of the authorized program. During verification of the authorized program 152 the higher-level component 150 will compute the modification detection code for the program, and then, using the stored public-key 160, it will verify that the digital signature as stored with the authorized software 154 is correct.

Id., at 19:29-44.[20]

---

[18] I understand that the term "first non-volatile memory area of the computer" has been given its plain and ordinary meaning by the Court. Dkt. 69 at 5.
[19] The Court construed "set up a verification structure" to mean "establishing or certifying the existence of a pseudo-unique key and establishing at least one license-record location," and I explained above how that limitation is disclosed in the Jablon Patent.
[20] I note that the Jablon Patent here alternately refers to the "higher level component" using the identifying numeral 150 or 160. Because Figure 7 uses the numeral 150 to refer to the higher

486. Under the claim scope that Ancora affords to this claim limitation, it is disclosed by either the client software or the server software disclosed in the Jablon Patent, working independently or together, for the reasons described herein as well as in limitation 1[b], above.

2. Limitation 7[b] is Disclosed by the Jablon Patent

487. Limitation 7[b] of the '941 patent recites:

> establishing at least one license-record location in the first nonvolatile memory area or in the erasable, nonvolatile memory area of the BIOS

488. The Jablon Patent also discloses establishing at least one license-record location in the first nonvolatile memory area or in the erasable, nonvolatile memory area of the BIOS.

489. I understand that the Court has indicated that "establishing at least one license-record location" may include "forming a license-record by at least partially encrypting the contents used to form a license-record with other predetermined data contents, using at least part of the pseudo-unique key; and storing the encrypted license-record." I further understand the Court indicated that this information is not for the jury to consider. See Dkt. 69 at 4. However, I understand that because the Court has determined as a matter of law that the claim scope includes such an operation, that evidence that the Jablon Patent discloses that operation shows that the Jablon Patent discloses "establishing at least one license record location."

490. The Jablon Patent discloses forming a license record, *e.g.* the public-key cryptosystem-signed modification detection code, by partially encrypting the contents used to form a license record with other predetermined data contents, including part of a pseudo-unique key, under the meaning ascribed to the term by Ancora in its infringement contentions. Specifically, the Jablon Patent discloses that the MDC is digitally signed using the trusted

level component, a POSITA would have understood that the reference to "higher level component 160" is a simple typographical error that was meant to refer to higher level component 150.

authority's private key, which a POSITA would have understood comprises encryption of the contents of the MDC. *Id.* at 19:29-44. A POSITA would have had this understanding because the Jablon Patent is explicit that its digital signature can be generated, for example, using an RSA digital signature, which is the same technique described for the Arbaugh Patent and Arbaugh Provisional, above:

> Modification detection codes are also commonly used in conjunction with the use of "public-key digital signatures", which can authenticate the originator of a message. Creating a digital signature for a message often involves computing a modification detection code for the message, and then a further computation that "signs" the code with a private key held only by the originator of a message. A public-key that corresponds to the originator's private key is made widely available. The signature can be then be verified by any person who has access to the originator's public-key, with a computation that uses the modification detection code, the signature, and the public-key. The digital signature technique, a popular example of which is described in U.S. Pat. No. 4,405,829 ("RSA"), is used in an enhanced form of our invention.

Id. at 5:53-68. As I explained earlier, this digital signature is generated using a form of encryption technique where a private key is used to apply a data transformation to the data to be signed, such that the original data can only be recovered by a party using the public key that corresponds to the private key. A POSITA would have understood that the application of a technique that reversibly converts data into a different set of data that can only be recovered by using the appropriate public key is a form of encryption, and that digital signatures such as in the Arbaugh Patent and in the Jablon Patent therefore disclose this limitation.

491. Under the claim scope that Ancora affords to this claim limitation, it is disclosed by either the client software or the server software disclosed in the Jablon Patent, working independently or together, for the reasons described herein as well as in limitation 1[b], above.

F. Claim 8 is Anticipated by the Jablon Patent

1. __Limitation 8[a] is Disclosed by the Jablon Patent__

492. Limitation 8[a] of the '941 patent recites:

A method according to claim 6 wherein establishing a license- record includes the steps of: forming a license-record by encrypting of the contents used to form a license-record with other predetermined data contents, using the key.

493. For the preceding limitation I explained why the limitation was met under the Court's note that "establishing at least one license-record location" may include "forming a license-record by at least partially encrypting the contents used to form a license-record with other predetermined data contents, using at least part of the pseudo-unique key; and storing the encrypted license-record." I then explained why the Jablon Patent discloses that specific circumstances. Because this element is entirely subsumed by the analysis above for claim 7, this element is disclosed by the Jablon Patent for the same reasons as claim 7.

494. Under the claim scope that Ancora affords to this claim limitation, it is disclosed by either the client software or the server software disclosed in the Jablon Patent, working independently or together, for the reasons described herein as well as in limitation 1[b], above.

2. <u>Limitation 8[b] is Disclosed by the Jablon Patent</u>

495. Limitation 8[b] of the '941 patent recites:

establishing the encrypted license-record in one of the at least one established license-record locations.

496. The encrypted license record identified above is stored in one of the established license-record locations. As discussed earlier for claim 7, the license-record location is the location where the license-record is stored. In the Jablon Patent, the license-record is stored along with each of the relevant programs, as shown in Figure 7 of the Jablon Patent and explained in the appropriate

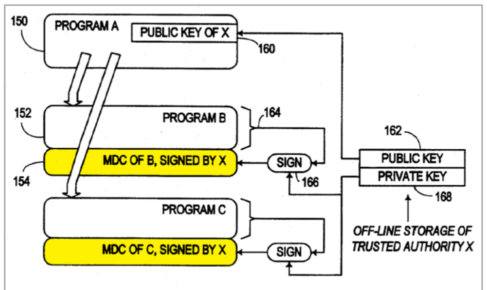


FIG. 7

accompanying text (right) (*Id.* at Fig. 7 (originally annotated by Dr. Nettles)). As explained in the Jablon Patent, the license records (in this case, the MDC signed using the private key of trusted authority X) is appended to the program file to be verified:

> During configuration of an authorized program 152, a modification detection code is first computed 164, as before. But then a public- key digital signature is constructed for this modification detection code 166, using the private-key of a trusted authority 168, and this signature is stored associated with the authorized software 154, possibly appended to the end of the file.

Id., at 19:29-35.

497. Under the claim scope that Ancora appears to afford to this claim limitation, it is disclosed by either the client software or the server software disclosed in the Jablon Patent, working independently or together, for the reasons described herein as well as in limitation 1[b], above.

G. Claim 9 is Anticipated by the Jablon Patent

498. The Jablon Patent renders claim 9 invalid because it discloses all of the limitations discussed above for claim 7, from which claim 9 depends, and also discloses the additional limitations in dependent claim 9.

1. __Limitation 9[a] is Disclosed by the Jablon Patent__

499. Limitation 9[a] of the '941 patent recites:

> A method according to claim 7 wherein verifying the program includes the steps of: encrypting the licensed-software-program's license-record contents from the volatile memory area or decrypting the license-record in the erasable, nonvolatile memory area of the BIOS, using the pseudo-unique key

500. The Jablon Patent discloses this limitation at least in its disclosure of verifying the digital signature of the MDC, which constitutes "decrypting the license-record in the erasable, nonvolatile memory area of the BIOS, using the pseudo-unique key." The Jablon Patent discloses using RSA digital signatures to verify that the MDC was digitally signed by the trusted

501. authority. As described above with respect to the Arbaugh Provisional, an RSA

digital signature constitutes an "encryption" and verifying an RSA digital signature constitutes a "decryption," as was well-known to those having ordinary skill in the art and is confirmed by U.S. Pat. No. 4,405,829, which is identified as "RSA" and referenced in the Jablon Patent at 5:67.

502. Jablon Discloses that decryption occurs using the public key that is stored in the higher level component 150, which constitutes a pseudo-unique key at least under Ancora's infringement contentions and its interpretation of the claims, as I discussed above with respect to the Arbaugh Patent. Moreover, the license record is stored in the erasable, non-volatile memory area of the BIOS, as it is stored in write-protectable non-volatile memory 94 in the Jablon Patent. *Id.*, at Fig. 5, 19:35-55.

2. __Limitation 9[b] is Disclosed by the Jablon Patent__

503. Limitation 9[b] of the '941 patent recites:

> comparing the encrypted licenses-software-program's license- record contents with the encrypted license-record in the erasable, nonvolatile 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.

504. Ancora in its infringement contentions has effectively alleged that a hash of a software component is part of the content of that software component.

505. Ancora in its infringement contentions has effectively alleged that a hash of a software component is part of the content of that software component. See Ancora Final Infringement Contentions Ex. B, at 130-142. In particular, Ancora alleged that the hash carried in the VB Meta data structure is content of the software component. *Id.* Based on such a broad interpretation of "license-software-program's license record contents" as recited in claim 9 (which interpretation I do not agree with), a POSITA would have understood that the hash or MDC of a software component is content of the software component, which satisfies the claim

element.

506. Under this broad interpretation, Jablon discloses this limitation at least in its disclosure of verifying the digital signature of the MDC, which constitutes "comparing the license-software program's license-record contents with the decrypted license-record in erasable non-volatile memory area of the BIOS." Jablon describes the verification process as including a comparison between a computed MDC and a stored, digital signed MDC. *Id.*, at 19:28-44. The MDC is computed based on the software component similar to a hash. *Id.*, at 5:25-51.

507. The Jablon Patent explains the use of modification detection codes involves comparing a modification detection code using a known technique, and then comparing it to a pre-computed stored code, and only running a program if the two codes are "identical":

> In one variation, modification detection codes provide a suitably strong means of integrity verification. Before the first program transfers control to the second program, it computes, using known techniques, a modification detection code for the second program, and compares it to a pre-computed stored code in protectable non- volatile memory. If the codes are not identical, then the second program is not run, and the user is warned of the problem.

Id., at 8:60-68. This approach is also used in the embodiment that I rely on for my opinion here, where the modification detection code is first retrieved and verified using a digital signature verification process, and if the signature is verified, comparing the computed code to the now-unsigned code:

> The verification process occurs in two steps: first the signed modification detection code is retrieved during the "unsigning" or digital signature verification process. Then, if the signature verification was successful, the unsigned code is compared to the computed code on the loaded program. The program is not run if either the signature verification fails, or if the code is incorrect.

Id., at 9:14-21. As mentioned for the previous limitation, the license record is stored in the erasable non-volatile memory, as it is stored in "protected non-volatile memory Fig. 5-94," which is explained in Figure 5 to be a "write-protectable non-volatile memory." *Id.*, at 19:50-55,

Fig. 5. A POSITA would have readily understood that a write-protectable memory is one that can be, but need not be, write-protected. That is to say, it may be protected from writing, or it may not be, and that if it is not protected from being written, it is erasable.

H. Claim 10 is Anticipated by the Jablon Patent

508. Claim 10 of the '941 patent recites:

> 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.

509. The Jablon Patent states that if the comparison between a computed modification detection code and a stored modification code for a program does not reveal the codes to be "identical," then the program will not be permitted to execute. *Id.*, at 8:66-68 ("If the codes are not identical, then the second program is not run, and the user is warned of the problem."); 9:18-21 ("Then, if the signature verification was successful, the unsigned code is compared to the computed code on the loaded program. The program is not run if either the signature verification fails, or if the code is incorrect."); 19:39-44 ("During verification of the authorized program 152 the higher-level component 150 will compute the modification detection code for the program, and then, using the stored public-key 160, it will verify that the digital signature as stored with the authorized software 154 is correct.")

510. A POSITA would have considered the step of not running the program to be restricting the program's operation, because the program is not permitted to operate. This limitation is "predetermined" because the Jablon Patent explains that when the comparison fails to verify the integrity of a program, the program does not execute. The system must be configured to impose this restriction, and a POSITA would have therefore understood that this configuration would be made prior to the verification failure. Lastly, the comparison in the Jablon Patent discloses that the program will not be executed when there is "non-unity" or

"insufficiency" in the comparison. The Jablon Patent discloses that unless the computed modification detection code and the decrypted modification detection code are "identical," the comparison will be deemed a verification failure and the program will not execute. *Id.*, at 8:66-68. Similarly, if the verification code is "incorrect," the comparison will be deemed a verification failure. *Id.*, at 9:19-20. Two codes not being identical would have been considered by a POSITA to be "non-unity," and a code being incorrect would have been considered by a POSITA to be "insufficiency."

I. Claim 11 is Anticipated by the Jablon Patent

511. Claim 11 of the '941 patent recites:

> A method according to claim 1 wherein the volatile memory is a RAM.

512. The Jablon Patent renders claim 11 invalid because it discloses all of the limitations discussed above for claim 1, from which claim 11 depends, and also discloses that "the volatile memory is a RAM" in dependent claim 10.

513. The Jablon Patent discloses that the computer system has a random-access memory, RAM 13. *Id.* at 13:45-46 ("RAM 13 and ROM 15 are connected to the address bus 12 and data bus 14 to allow access by the CPU 36."). I note that claim 11 does not impose any other requirements for this limitation other than that the claimed computer has a volatile memory that is a RAM, which is disclosed by the Jablon Patent.

J. Claim 12 is Anticipated by the Jablon Patent

514. Claim 12 of the '941 patent recites:

> The method of claim 1, wherein a pseudo-unique key is stored in the non-volatile memory of the BIOS.

515. The Jablon Patent renders claim 12 invalid because it discloses all of the limitations discussed above for claim 1, from which claim 12 depends, and also discloses that "a

pseudo-unique key is stored in the non-volatile memory of the BIOS" in dependent claim 12.

516. The Jablon Patent discloses that a pseudo-unique key is stored in the non-volatile memory of the BIOS in the form of a public key of the trusted authority that is stored in the BIOS. As Jablon explains, the public key is stored in the non-volatile memory of the BIOS:

> The corresponding public-key of this authority 162 is then stored in the higher-level component 160, rather than storing the modification detection code of the authorized program.

Id., at 19:35-39. As explained in the Jablon Patent, and as shown in Figure 5, the public key is stored in the write-protectable non-volatile memory 94. *Id.*, at 19:50-52. This public key is a pseudo-unique key for the same reasons I explained above with respect to the Arbaugh Patent, and at least for the reason that the plaintiff and patent-owner in this case, Ancora, has argued that a public key used to decode data encrypted with a private key constitutes a "pseudo-unique key."

K. Claim 13 is Anticipated by the Jablon Patent

517. Claim 13 of the '941 patent recites:

> The method of claim 1, wherein a unique key is stored in a first non-volatile memory area of the computer.

518. The Jablon Patent renders claim 13 invalid because it discloses all of the limitations discussed above for claim 1, from which claim 13 depends, and also discloses that "a unique key is stored in a first nonvolatile memory area of the computer" in dependent claim 13. In light of Ancora's infringement contentions and interpretation of claims, the public key in the Jablon Patent constitutes a "unique key" for the same reason that the public key in the Arbaugh Patent constitutes a "unique key."

L. Claim 14 is Anticipated by the Jablon Patent

519. Claim 14 of the '941 patent recites:

> 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.

520. The Jablon Patent renders claim 14 invalid because it discloses all of the limitations discussed above for claim 13, from which claim 14 depends, and also discloses that "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" in dependent claim 14.

521. Ancora in its infringement contentions has effectively alleged that a hash of a software component is part of the content of that software component. *See* Ancora Final Infringement Contentions Ex. B, at 153-161. In particular, Ancora alleged that the hash carried in the VB Meta data structure is content of the software component. *Id.* Based on such a broad interpretation of "encrypting a license record data in the program" as recited in claim 14 (which interpretation I do not agree with), a POSITA would have understood that the hash of a software component is content of the software component. Under such a broad interpretation, the Jablon Patent's digital signature of the MDC satisfies the "encrypting a license record data in the program" element, because the MDC is computed from the software component like a hash. Jablon Patent, 5:26-52.

522. The Jablon Patent discloses that when the verification record is being set up, the unique key is used to encrypt license record data in the program. The key disclosed in the Jablon Patent constitutes a unique key under Ancora's infringement contentions and interpretation of claims. In the Jablon Patent, the unique key used to encrypt the license record data is the private key belonging to the trusted authority.[21]

[21] As indicated with respect to the analysis for the Arbaugh Patent, Ancora here does not identify in its infringement contentions any unique key that is stored in the non-volatile memory of the computer system, as this claim requires. To the extent that Ancora later argues that a key present

> During configuration of an authorized program 152, a modification detection code is first computed 164, as before. But then a public- key digital signature is constructed for this modification detection code 166, using the private-key of a trusted authority 168, and this signature is stored associated with the authorized software 154, possibly appended to the end of the file.

Id., at 19:29-35. A POSITA would have understood that when the Jablon Patent refers to a public-key digital signature being "constructed for this modification detection code 166," it is referring to digitally signing the modification detection code. *Id.* This uses a private key to sign the modification detection code. *Id.* at 9:11-13 ("This signing process generally occurs off-line, on a more trustworthy machine, in order to keep the private key secret."). Signing techniques were well-known in the art, and the Jablon Patent specifically identifies one well-known way using a technique devised by the security research firm RSA:

> Creating a digital signature for a message often involves computing a modification detection code for the message, and then a further computation that "signs" the code with a private key held only by the originator of a message. A public-key that corresponds to the originator's private key is made widely available. The signature can be then be verified by any person who has access to the originator's public-key, with a computation that uses the modification detection code, the signature, and the public-key. The digital signature technique, a popular example of which is described in U.S. Pat. No. 4,405,829 ("RSA", is used in an enhanced form of our invention.

Id., at 5:56-68. Jablon therefore discloses this limitation because the license record data is encrypted using this digital signature process and then is placed, in one embodiment, with the program data. *Id.* at 19:33-35 ("this signature is stored associated with the authorized software 154, possibly appended to the end of the file.").

M. Claim 16 is Anticipated by the Jablon Patent

523. Claim 16 of the '941 patent recites

> The method according to claim 13, wherein the step of verifying the program includes a decrypting the license record data accommodated in the erasable

in the non-volatile memory of the computer meets this limitation, I reserve the right to respond to that argument.

second non-volatile memory area of the BIOS using at least the unique key."

524. The Jablon Patent renders claim 16 invalid because it discloses all of the limitations discussed above for claim 13, from which claim 16 depends, and also discloses "decrypting the license record data accommodated in the erasable second non-volatile memory area of the BIOS using at least the unique key" in dependent claim 16.

525. The Jablon Patent discloses that the license record data is in the erasable second non-volatile memory area of the BIOS as described above with reference to claim 1(c). The license record data includes the digitally-signed modification detection code and, as such, the disclosure in the Jablon Patent relating to the verification of program integrity using the digitally-signed MDC is relevant to this limitation:

> The verification process occurs in two steps: first the signed modification detection code is retrieved during the "unsigning" or digital signature verification process. Then, if the signature verification was successful, the unsigned code is compared to the computed code on the loaded program.

Id., at 9:14-19. The use of a digital-signature technique, like the exemplary RSA public key cryptography system described in the Jablon Patent, requires "decrypting" the digitally- signed MDC using the public key of trusted authority that originally signed the MDC. *Id.*, at 5:55-65. As described above for claim 13, the public key stored in the computer system in the Jablon Patent discloses the claimed "unique key," at least insofar as Ancora has asserted that storing a public key in a nonvolatile memory in a device meets this limitation.

526. I note that the private key used to generate the cryptographic signature is not the key stored in the memory in the Jablon Patent. Instead, it is the public key portion of the asymmetric key pair (private key and public key) which is stored in the memory in the Jablon Patent. Ancora contends that the limitation of "encrypting a license record data in the program using at least the unique key" is met by a public key stored on an accused device

XVII. OVERVIEW OF THE CHOU PATENT (U.S. PATENT NO. 5,892,906)

527. The Chou patent is directed to discouraging computer theft. Chou Patent, at 2:10-It does so by using a security routine that is stored in BIOS memory, and a key device that is kept by the user and that can be mounted onto the computer. *Id.*, at 3:37-62. The BIOS memory is implemented in an EEPROM device, which is erasable non-volatile memory. *Id.* The key device stores a unique key device serial number and an encrypted value M, which represents the product of the key device serial number and the computer ID number. *Id.*, at 5:42-45. When the key device is connected to the computer, a public key stored in the BIOS memory is used to decrypt the encrypted value M. *Id.*, at 5:45-47. Then the computer ID is read from the BIOS memory. *Id.*, at 5:47-49. The key device serial number and the computer ID number are multiplied and compared to the decrypted value. *Id.*, at 5:49-62. If they match, then the user had been verified, and remaining boot code is executed. *Id.* If they do not, the user has not been verified and execution of boot code is terminated. *Id.*

528. The Chou patent discloses that "[i]nclusion of routines for executing a security function 25 with the BIOS routines is particularly useful in preventing a thief from bypassing

529. security measures which might have been implemented on the hard drive, or in an application program, or which previously made use of the CMOS RAM 17. Unless the BIOS routine has completely executed, the computer operating system can never be accessed rendering the computer inoperative." *Id.*, at 3:55-62. Thus, Chou discloses that sensitive security information, such as a public key used for verification, may be stored in BIOS memory. Storing the public key in this manner provides security benefits compared to storing it in an ordinary hard drive or an application program.

530. The Chou patent also discloses that a digital signature of the computer ID may also be generated and given to the user, which allows the user to access an administrative mode

of the computer if the user loses passwords. *Id.*, at 8:35-41.

XVIII. THE CHOU PATENT IS PRIOR ART

531. The Chou Patent was filed on July 19, 1996 and issued on April 6, 1999. The ’941 Patent was filed on October 1, 1998, claiming priority to an Israeli patent application filed on May 21, 1998. Therefore, I understand that the Chou Patent qualifies as prior art under 35 U.S.C. § 102(e) because it is a patent publication that issued from a patent application filed in the U.S. before the earliest conception date asserted by Ancora for the ’941 Patent.

XIX. THE COMBINATION OF THE ARBAUGH PATENT AND THE CHOU PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE ’941 PATENT

532. It is my opinion that the Arbaugh Patent in combination with the Chou Patent renders obvious claims 1-3, 6-14, and 16 of the ’941 Patent under 35 U.S.C. 103(a). Each limitation of claims 1-3, 6-14, and 16 of the ’941 Patent is also disclosed by the combination of the Arbaugh Patent and the Chou Patent, either literally or inherently, or is obvious in view of the Arbaugh Patent and the Chou Patent.

533. As discussed above in Section XIII.A, the Arbaugh Patent discloses at least all expressly recited elements of claim 1 of the ’941 patent. As also discussed above, the Arbaugh Patent discloses that the cryptographic hash of a software component may be extracted from a digital signature. Arbaugh Patent, 14:33-45; 15:17-48. As discussed, for example, regarding claim 16, the Arbaugh Patent discloses to one of ordinary skill in the art that such decryption may be carried out using a public key. *Id.*, at 15:17-48. The Chou Patent discloses to one of ordinary skill in the art that a public key used for security-related purposes (such as the use of the public key in the Arbaugh Patent) may be usefully stored in erasable BIOS memory, rather than in a hard drive or an application program. Chou Patent, at 3:52-62; Fig. 3. Storing such a security-related item in BIOS memory would prevent or render more difficult bypassing security

measures for protecting the security-related item from an unauthorized, possibly malicious, party. *Id.* It would have been obvious to one or ordinary skill in the art based on these disclosures to store the Arbaugh Patent's public key in erasable BIOS memory, as disclosed by the Chou Patent. Thus, the combination of the Arbaugh Patent and the Chou Patent discloses all elements of claim 1, including "establishing or certifying the existence of a pseudo-unique key and establishing at least one license-record location" (which is the Court's construction of the "set up a verification structure" element in claim 1.)

534. A person of ordinary skill in the art would have been motivated to combine the disclosures of the Arbaugh Patent and the Chou Patent as discussed above. The Arbaugh Patent discloses that a public key in the computer may be used to decrypt a digital signature for purposes of verifying integrity of bootloader code. Arbaugh Patent, 4:38-45. Such verification prevents malicious parties from accessing and using the computer without authorization. *Id.*, at 1:46-52; Abstract. The Chou Patent is also directed to ensuring that only an authorized user can access and use a computer, and that a thief, who is not authorized, cannot access or use the computer. Chou Patent, 2:10-33; 3:55-62. The Chou Patent discloses that this goal can be achieved by storing information used to ensure security, such as the security routine and the public key used to verify the authorized user of the computer, in erasable BIOS memory. *Id.*, at 3:52-62. Thus, a person of ordinary skill in the art would have been motivated to combine these disclosures of the Arbaugh Patent and the Chou Patent. In making the combination, a person of ordinary will use a known technique (*i.e.*, storage of a public key in erasable BIOS memory as disclosed by the Chou Patent) to improve a similar method (*i.e.*, use of a public key for purposes of preventing unauthorized access and use of a computer as disclosed in the Arbaugh Patent) in the same way, and would have a reasonable expectation of success.

535. Moreover, the Chou Patent discloses security benefits that arise from storing security-related information in erasable BIOS memory. The Arbaugh Patent is concerned with ensuring security of the boot-up sequence, among other things, of a computer based on digital signatures of software components and the public keys used to decrypt them. Based on the Chou Patent's disclosure, it would have been obvious to a person of ordinary skill to store the security-related information of the Arbaugh Patent, such as the public key, also in erasable BIOS memory. A person of ordinary skill would rely upon a known technique (the Chou Patent's disclosure of storing security-related information in erasable BIOS memory) to a known device (the Arbaugh Patent computer, which is configured to carry out secure boot verification based on digital signatures and the public key) that ready for improvement to achieve a predictable result (more secure storage of the public key in erasable BIOS memory.)

536. Additionally, in a computer system like that of Arbaugh that is based on the PC architecture, there are a small number of possibilities for storing the public key in a non-volatile manner on the computer. The Arbaugh Patent discloses PROMs, ROM, and the hard disk as possibilities. Based on the Chou Patent's disclosure that the security-related information could usefully be stored in erasable BIOS memory, it would have been obvious for one of ordinary skill in the art to try Chou Patent's storage method for the public key in the Arbaugh Patent.

537. A person of ordinary skill in the art would have a reasonable expectation of success in modifying the system of the Arbaugh Patent based on the disclosure of the Chou Patent. As discussed above, Chou expressly discloses storing security-data in erasable BIOS memory. Based on this disclosure one of ordinary skill in the art could modify Arbaugh (to the extent necessary) to implement storage of public keys in such erasable BIOS memory. In particular, flash memory for enabling such functionality were well known at the relevant time, as

is apparent from the disclosures of the Arbaugh Patent and the Chou Patent.

XX. THE COMBINATION OF THE JABLON PATENT AND THE CHOU PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE ’941 PATENT

538. It is my opinion that the Jablon Patent in combination with the Chou Patent renders obvious claims 1-3, 6-14, and 16 of the ’941 Patent under 35 U.S.C. 103(a). Each limitation of claims 1-3, 6-14, and 16 of the ’941 Patent is also disclosed by the combination of the Jablon Patent and the Chou Patent, either literally or inherently, or is obvious in view of the Jablon Patent and the Chou Patent.

539. As discussed above in Section XVI.A., the Jablon Patent discloses or suggests at least all expressly recited elements of claim 1 of the ’941 patent. As discussed earlier, the Jablon Patent discloses an embodiment in which its method for ensuring integrity of a program run on the computer is based on a digital signature of the program. Jablon Patent, at 19:2-5. In particular, the program that is to be run is first verified against that digital signature. *Id.*, at 19:39-44. The Jablon Patent discloses that such verification enhances security. *Id.*, at 1:10-18.

540. The Chou Patent suggests to one of ordinary skill in the art and discloses that computer-security related routines may be usefully stored in erasable BIOS memory. Chou Patent, at 3:15-62. Additionally, security-related data, such as a public key and the computer I.D., that are used for purposes of verification of authorization to use the computer may also be stored in the erasable BIOS memory. *Id.*, at 4:6-19. Storing such security-related items in BIOS memory would prevent or render more difficult bypassing security measures for protecting the security-related item from an unauthorized, possibly malicious, party. *Id.*, at 3:15-62.

541. Based on these disclosures, it would have been obvious to one or ordinary skill in the art to store the digital signature in the Jablon Patent (which corresponds to the recited “verification structure”) in erasable BIOS memory, as disclosed by the Chou Patent. Thus, the

combination of the Jablon Patent and the Chou Patent discloses all elements of claim 1, including "set[ting] up a verification structure . . . in the erasable, non-volatile memory of the BIOS."

542. A person of ordinary skill in the art would have been motivated to combine the Jablon Patent and the Chou Patent as discussed above. The Jablon Patent discloses that the digital signature may be used to verify the integrity of a software program that is to be run on a computer. Jablon Patent, at 15:53-67, 9:1-54, and 8:64-19:62. Such verification enhances system security. *Id.* Chou is also directed to ensuring security by permitting only an authorized user to access and use a computer, and preventing an unauthorized party such as a thief from accessing or using the computer. Chou Patent, 2:10-33; 3:55-62. Chou discloses that this goal can be achieved by storing information used to ensure security, such as security routines, and a public key and computer I.D. that are used to verify the authorized user of the computer, in erasable BIOS memory. *Id.*, at 3:52-62. A person of ordinary skill in the art would have been motivated to combine the disclosures of the Jablon Patent and the Chou Patent by storing the Jablon Patent's digital signature in erasable BIOS memory, as disclosed by the Chou Patent. In making the combination, a person of ordinary skill would use of a known technique (*i.e.*, storage of security-related information in erasable BIOS memory as disclosed by the Chou Patent) to a known device ready for improvement (*i.e.*, the Jablon Patent's scheme for ensuring integrity and enhancing security based on verifying a software program using a digital signature) to yield a predictable result (*i.e.*, storage of the Jablon Patent's digital signature in BIOS memory).

543. Additionally, in an IBM-compatible PC like that of the Jablon Patent, there are only a small number of possibilities for storing the digital signature on the computer. The Jablon Patent discloses ROM, protectable non-volatile memory, and the hard disk as possibilities. Jablon Patent, at 11:4-20, Fig. 1. Based on Chou's disclosure that security-related information

could usefully be stored in erasable BIOS memory, it would have been obvious for one of ordinary skill in the art to try Chou's storage method for the digital signature of Jablon.

544. A person of ordinary skill in the art would have a reasonable expectation of success in modifying the system of the Jablon Patent based on the disclosure of the Chou Patent. This is because, as discussed above, Chou expressly discloses storing security-data in erasable BIOS memory. Based on this disclosure one of ordinary skill in the art could modify the Jablon Patent (to the extent necessary) to implement storage of digital signatures in such erasable BIOS memory. In particular, flash memory for enabling such functionality were well known at the relevant time, as is apparent from the disclosures of the Jablon Patent and the Chou Patent.

XXI. OVERVIEW OF THE MIROV PATENT (U.S. PATENT NO. 6,138,236)

545. The Mirov Patent is directed to authenticating firmware in a manner that permits the ability to upgrade the firmware without compromising the integrity of the firmware. Mirov Patent, at 2:7-10. Start-up instructions (*i.e.*, BIOS) for starting up the computer are stored in flash PROM. *Id.*, at 3:56-64. Part of these start-up instructions, the secure micro-code, are stored in an authentication section of the PROM that is read-only and cannot be changed. *Id.* The other part of these instructions, the unsecured micro-code, are stored in a programmable section of the flash PROM, which permits them to be upgraded. *Id.* The programmable section of the flash PROM also stores a digital signature of the programmable section. *Id.*, at 4:4-6; 4:18-41. When the computer is turned on, the secure micro-code decrypts the digital signature using a public key stored in the authentication section of PROM to obtain a verification hash value. *Id.*, at 3:66-4:17; 4:26-41. It also calculates a data hash value by applying the relevant hash algorithm to the unsecured micro-code. *Id.* If the verification hash value matches the data hash value, then the unsecured micro-code has been verified and will be permitted to execute. *Id.*, at 4:18-26. If there is no match, then the unsecured micro-code is unverified and will not be permitted to execute.

Id., at 4:18-26; 4:66-5:14; Fig. 4. In this case, the user may be provided an error message and a recovery solution for the user to obtain valid unsecured micro-code. *Id.*, at 4:66-5:14; Fig. 4.

XXII. THE MIROV PATENT IS PRIOR ART

546. The Mirov Patent was filed on July 1, 1996 and issued on October 24, 2000. The ’941 Patent was filed on October 1, 1998, claiming priority to an Israeli patent application filed on May 21, 1998. Therefore, I understand that the Mirov Patent qualifies as prior art under 35 U.S.C. § 102(e) because it is a patent publication that issued from a patent application filed in the U.S. before the earliest conception date asserted by Ancora for the ’941 Patent.

XXIII. THE COMBINATION OF THE ARBAUGH PATENT AND THE MIROV PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE ’941 PATENT

547. It is my opinion that the Arbaugh Patent in combination with the Mirov Patent renders obvious claims 1-3, 6-14, and 16 of the ’941 Patent under 35 U.S.C. 103(a). Each limitation of claims 1-3, 6-14, and 16 of the ’941 Patent is also disclosed by the combination of the Arbaugh Patent and the Mirov Patent, either literally or inherently, or is obvious in view of the Arbaugh Patent and the Mirov Patent.

548. As discussed above in Section XIII.A, the Arbaugh Patent discloses at least all expressly recited elements of claim 1 of the ’941 patent. As also discussed above, the Arbaugh Patent discloses that the cryptographic hash of a software component may be extracted from a digital signature in a secure boot process. Arbaugh Patent, 14:33-45; 15:17-48. As discussed, for example, regarding claim 16, the Arbaugh Patent discloses to one of ordinary skill in the art that such decryption may be carried out using a public key. *Id.*, at 15:17-48. The Mirov Patent also discloses a secure boot process and indicates that the digital signatures of software components to be verified and the public key used to decrypt them may be stored in flash PROM that stores micro-code for starting up a computer, *i.e.*, BIOS. *Id.*, at 3:56-4:54. It would have been obvious

to one of ordinary skill in the art based on these disclosures to store the Arbaugh Patent's public key in flash PROM that stores BIOS, as disclosed by the Mirov Patent. Thus, the combination of the Arbaugh Patent and the Mirov Patent discloses all elements of claim 1, including "establishing or certifying the existence of a pseudo-unique key and establishing at least one license-record location" (which is the Court's construction of the "set up a verification structure" element in claim 1.)

549. A person of ordinary skill in the art would have been motivated to combine the disclosures of the Arbaugh Patent and the Mirov Patent as discussed above. The Arbaugh Patent discloses a secure boot process in which a public key in the computer may be used to decrypt a digital signature for purposes of verifying integrity of bootloader code. Arbaugh Patent, 4:38-45. Such verification prevents malicious parties from accessing and using the computer without authorization. *Id.*, at 1:46-52; Abstract. The Mirov Patent is also directed to ensure a secure boot process in a context in which the boot code may be upgraded. Mirov Patent, 2:7-20; 3:55- 4:55. Like the Arbaugh Patent, the Mirov Patent also discloses that a public key in the computer may be used to decrypt a digital signature for purposes of verifying the integrity of bootloader code. *Id.* Thus, a person of ordinary skill in the art would have been motivated to combine these disclosures of the Arbaugh Patent and the Mirov Patent. In making the combination, a person of ordinary skill will use a known technique (*i.e.*, storage of a public key in flash PROM memory that also stores BIOS code, as disclosed by the Mirov Patent) to improve a similar method (*i.e.*, use of the public key for verification of a digital signature of bootloader code during the secure boot process, as disclosed in the Arbaugh Patent) in the same way, and will have a reasonable expectation of success.

550. Additionally, in a computer system like that of the Arbaugh Patent that is based

on the PC architecture, there are a small number of possibilities regarding where to store the public key in a non-volatile manner on the computer. The Arbaugh Patent discloses PROMs, ROM, and the hard disk as possibilities. Like the Arbaugh Patent, Mirov discloses a secure boot-up process. Given the similarities between the two systems and benefits of the technique disclosed in the Mirov Patent, one of ordinary skill in the art would have been motivated to implement a storage solution in the Mirov Patent in the system of the Arbaugh Patent. For example, it would have been obvious for one of ordinary skill in the art to store the Arbaugh Patent's public key in flash PROM memory that also stores BIOS code, as disclosed by Mirov .

551. A person of ordinary skill in the art would have a reasonable expectation of success in modifying the system of the Arbaugh Patent based on the Mirov Patent's disclosure. Mirov expressly discloses storing security-data such as digital signatures and public keys in an erasable BIOS memory chip. Based on this disclosure one of ordinary skill in the art could modify Arbaugh (to the extent necessary) to implement storage of public keys in such erasable BIOS memory. In particular, flash memory for enabling such functionality were well known at the relevant time, as is apparent from the disclosures of the Arbaugh Patent and the Mirov Patent.

XXIV. THE COMBINATION OF THE JABLON PATENT AND THE MIROV PATENT INVALIDATES ALL ASSERTED CLAIMS OF THE '941 PATENT

552. It is my opinion that the Jablon Patent in combination with the Mirov Patent renders obvious claims 1-3, 6-14, and 16 of the '941 Patent under 35 U.S.C. 103(a). Each limitation of claims 1-3, 6-14, and 16 of the '941 Patent is also disclosed by the combination of the Jablon Patent and the Mirov Patent, either literally or inherently, or is obvious in view of the Jablon Patent and the Mirov Patent.

553. As discussed above in Section XVI.A., the Jablon Patent discloses or suggests at least all expressly recited elements of claim 1 of the '941 patent. As discussed earlier, the Jablon

Patent discloses an embodiment in which its method for ensuring integrity of a program run on the computer is based on a digital signature of the program. Jablon Patent, at 19:2-5. In particular, the program that is to be run is first verified against that digital signature. *Id.*, at 19:39-44. The Jablon Patent discloses that such verification enhances security. *Id.*, at 1:10-18.

554. The Mirov Patent also discloses a method of ensuring integrity of programs run on the computer (and, in particular, micro-code for starting up the computer, *i.e.*, BIOS) based on the digital signatures of the programs. The Mirov Patent discloses that the digital signatures may be stored in flash PROM that stores the micro-code for starting up the computer, *i.e.*, BIOS. Mirov Patent, at 3:56-4:54. It would have been obvious to one or ordinary skill in the art based on these disclosures to store the Jablon Patent's digital signature in flash PROM that stores BIOS, as disclosed by the Mirov Patent. Thus, the combination of the Jablon Patent and the Mirov Patent discloses all elements of claim 1, including "set[ting] up a verification structure . . . in the erasable, non-volatile memory of the BIOS."

555. A person of ordinary skill in the art would have been motivated to combine the disclosures of the Jablon Patent and the Mirov Patent as discussed above. The Jablon Patent discloses a method for ensuring integrity of a computer program that is run on a computer by verifying the program against a digital signature associated with the program. Jablon Patent, 19:2-5, 19:39-44. Such verification enhances security. *Id.*, at 1:10-18. The Mirov Patent is also directed to ensuring integrity and security of a computer program based on its digital signature. Mirov Patent, 2:7-20; 3:55-4:55. Given the similarity of purpose and functionality, a person of ordinary skill in the art would have been motivated to combine disclosures of the Jablon Patent and the Mirov Patent. In making the combination, a person of ordinary skill would use a known technique (*i.e.*, storage of the digital signature for ensuring integrity of a program in flash PROM

memory that also stores BIOS code, as disclosed by the Mirov Patent) to improve a similar method (*i.e.*, storage of the digital signature for ensuring authentication of a trusted source/authority and integrity of a program, as disclosed in the Jablon Patent) in the same way and would have a reasonable expectation of success.

556. Additionally, in an IBM-compatible PC like that of the Jablon Patent, there are only a small number of possibilities for storing the digital signature on the computer. The Jablon Patent discloses ROM, protectable non-volatile memory, and the hard disk as possibilities. Jablon Patent, 11:4-20, Fig. 1. Like the Jablon Patent, the Mirov Patent discloses a method for achieving integrity of a computer program by verifying it against its digital signature. Given the similarities in functionality of the two systems and the benefits of the method disclosed in the Mirov Patent, one of ordinary skill in the art would have been motivated to implement a storage solution in the Mirov Patent in the system of the Jablon Patent. For example, it would have been obvious for one of ordinary skill in the art to store the Jablon Patent's digital signature in flash PROM memory that also stores BIOS code, as disclosed by the Mirov Patent.

557. A person of ordinary skill in the art would have a reasonable expectation of success in modifying the system of the Jablon Patent based on the Mirov Patent's disclosure. The Mirov Patent expressly discloses storing security-data such as digital signatures and public keys in an erasable BIOS memory chip. Based on this disclosure one of ordinary skill in the art could modify the Jablon Patent (to the extent necessary) to implement storage of digital signatures in such erasable BIOS memory. In particular, flash memory for enabling such functionality were well known at the relevant time as is apparent from the disclosures of Jablon and Mirov.

XXV. OVERVIEW OF U.S. PATENT NO. 6,153,835 TO SCHWARTZ

558. U.S. Patent No. 6,153,835 to Schwartz ("Schwartz") was filed on June 7, 1995, claiming priority to an earlier U.S. patent application filed on October 14, 1993, and issued on

November 28, 2000. The ’941 Patent was filed on October 1, 1998, claiming priority to an Israeli patent application filed on May 21, 1998. Therefore, I understand that Schwartz qualifies as prior art under 35 U.S.C. § 102(e) because it is a patent that issued from a patent application filed in the U.S. before the priority date of the ’941 patent (i.e., May 21, 1988) or the earliest conception date in 1997 asserted by Ancora for the ’941 Patent.

559. Schwartz discloses an electronic postage scale system. Schwartz, Abstract; 1:25-27. The postage scale is controlled by an application program that may be updated from time to time. *Id.* at 10:15-20. During the update process, the system required a user to enter an authorization code to prove that the software was licensed. *Id.* at 10:21-26. Figure 8 of Schwartz depicts a general structure of the electronic components of the system (right). As shown in the figure, the system includes a memory section 250 and a ROM 213, which stores a unique serial number that was pre-assigned to the system. Schwartz, 7:48-50.

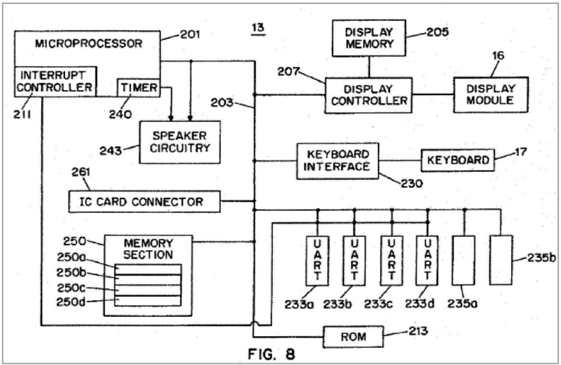


FIG. 8

560. The memory section has multiple sections 250a-d, including a flash EPROM section 250a and an erasable programmable read-only memory (EPROM) 250b. *Id.* at 7:50-54. As shown below in Figure 9, the system’s memory space includes a BIOS module 309, which was stored partially on the flash EEPROM 250a and partially on the EPROM

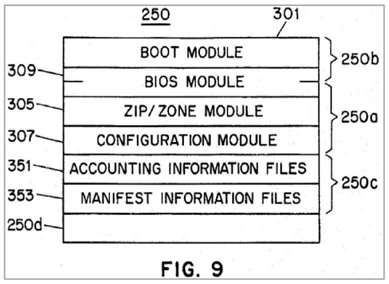


FIG. 9

250b. *Id.* at 8:16-20. Thus, as with the '941 patent, part of the BIOS was stored in a memory component that was difficult to modify (EPROM), while another part was stored in a memory component that was easier to modify (EEPROM).

561. When a new application program is installed on the system, a user is required to enter an authorization number. Schwartz, 10:21-25. The authorization number is based on information specific to the particular system, including the serial number and model number of the device and version numbers of the application program and various databases. *Id*. at 10:25-38. The system verifies the initial authorization by generating an electronic signature based on the same system information. *Id*. at 10:45-49. If the results match, the system stores the authorization information in a memory buffer that is part of flash EEPROM 250a. *Id*. at 10:47-54, 8:17-19. Subsequently, each time the system is operated, the system uses the stored signature to verify authorization by generating an electronic signature based on the same system information. *Id*. at 11:24-36. The system then compares the generated electronic signature against the electronic signature stored in the memory buffer (i.e., flash EEPROM 250a). *Id*. at 11:36-38. If the values match, the system begins to operate and the application program is executed. *Id*. at 11:38-40.

562. According to Schwartz, the verification method was useful to deter unauthorized copying because "even though the software [could] be copied onto [] similar systems, the latter [systems] would not be operational without proper authorization numbers, which need to be derived in part from their respective unique serial numbers." Schwartz, 12:30-41.

XXVI. OVERVIEW OF "USING SECURE COPROCESSORS," B. YEE, SCHOOL OF COMPUTER SCIENCE, CARNEGIE MELLON UNIVERSITY, 1994

563. "Using Secure Coprocessors," B. Yee, School Of Computer Science, Carnegie Mellon University, 1994 ("Yee") is a Ph.D. thesis published in 1994. The '941 Patent was filed

on October 1, 1998, claiming priority to an Israeli patent application filed on May 21, 1998. Regardless of the priority date or the earliest invention date claimed by Ancora, because Yee published more than one year before the earliest filing date of the '941 Patent in the United States, i.e., October 1, 1998, I understand that Yee qualifies as prior art to the '941 Patent on the basis of pre-AIA 35 U.S.C. § 102(b).

564. Yee is directed to the use of a secure coprocessor model in various computer and distributed applications. Yee at 5. Yee's threat model is directed to enhancing security for a computerized system by using cryptographic techniques that are implemented on a secure coprocessor of the system. Yee at 53-68. Yee discloses a number of applications for its models, including secure booting, copy protection, electronic currency, and postage meters. Yee at 13-34. In all cases, however, the applications share similar system architecture, cryptographic algorithms and protocols, bootstrap procedures, verification, and verification failure-related procedures. Yee at i-ii, 35-80.

565. A POSITA would have known that secure coprocessors offer a very strong level of security, as they are tamper-proof physical devices running security software. The cost of that added security is additional time, money, and effort to use the coprocessors. Nevertheless, Yee discloses several well-known cryptographic techniques that can be used effectively without a coprocessor.

566. Yee discloses non-volatile memory that stores a BIOS, but that also stores verification-related information, such as (i) digital signatures and cryptographic checksums, as well as (ii) encryption/decryption-related keys. Yee at 4, 38. Both aspects are used in Yee's applications, such as in connection with copy protection and postage meters. *Id.* at 19-21, 3133.

567. Yee discloses that the verification-related digital signatures and/or keys may be

provided from a server over the network, for example, in connection with system upgrades of a client device. Yee at 15, 20-21, 32.

568. Among commonly known cryptographic technique that Yee discloses is the concept of a key that is assigned to a client device for preserving security of information. For example, a server may assign a symmetric or asymmetric key to that client device for purposes of keeping communications or information secret. Yee at 20-21.

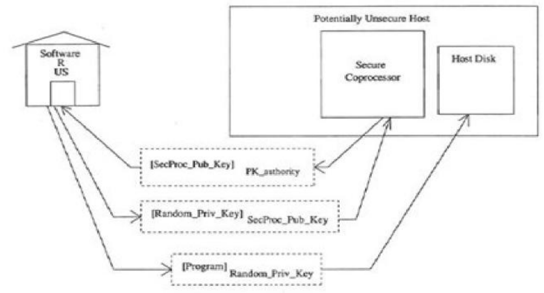

Potentially Unsecure Host
Software R US
Secure Coprocessor
Host Disk
[SecProc_Pub_Key] PK_authority
[Random_Priv_Key] SecProc_Pub_Key
[Program] Random_Priv_Key

Figure 3.1 Copy-Protected Software Distribution
The software retailer encrypts the copy-protected software with a random key. This key is encrypted using the public key of the secure coprocessor within the destination host, so only the secure coprocessor may decrypt and run the copy-protected software. The software retailer knows that the public key of the secure coprocessor is good, because it is digitally signed with the public key of the secure coprocessor distributor.

569. Cryptographic co-processors were known in the art at the time of the '941 patent, and they included at least ROM, RAM, and EEPROM. See Sudia at 13:32-64 ("Manufacture of trusted devices of the present invention is based on the presence of the following general features: (1) An embedded microprocessor [...] (2) An optional cryptographic coprocessor, which can perform standard mathematical encrypting and decrypting operations [...] (3) An input-output interface [...] (4) A memory subsystem that may potentially employ several types of memory storage technology, each having a different characteristics of permanence and accessibility, [...] ROM [...] ("EEPROM") or "FLASH" memory [...] RAM [...] for temporary calculations and temporary data storage but is lost when power is shut off."). *See also* Yee at 7 ("Several physically secure processors exist. [...] The µABYSS [103] and Citadel [105] systems

employ board-level protection.)", 8 ("Smartcards are another approach to physically secure coprocessing [54]. [...] newer smartcard designs such as some GEMPlus or Mondex cards [35] feature limited physical security protection, providing a true (simple) secure coprocessor."). A POSITA would have therefore known that many choices existed for adding stronger security mechanisms to any computer by incorporating it with a form of a secure or trusted coprocessor technology.

XI. SCHWARTZ IN COMBINATION WITH YEE INVALIDATES CLAIMS 1-2, 6-14, AND 16 OF THE '941 PATENT AS OBVIOUS

570. It is my opinion that Claims 1-2, 6-14, and 16 of the '941 patent would have been obvious over the combination of Schwartz and Yee. I discuss below how each limitation is disclosed in the combination of Schwartz and Yee.

571. As discussed above, Schwartz qualifies as prior art under pre-AIA 35 U.S.C. § 102(e) because it has an earlier filing date than the earliest date of invention for the '941 patent. Yee qualifies as prior art under pre-AIA 35 U.S.C. § 102(b) because it was published more than one year prior to the filing date of the '941 patent.

A. Claim 1 is Rendered Obvious by the Combination of Schwartz and Yee

1. __Limitation 1[preamble] Is Disclosed or Rendered Obvious by Schwartz__

572. Limitation 1[preamble] of the '941 patent recites:

> 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:

573. Schwartz discloses all elements of the preamble. Fig. 1 of Schwartz discloses an electronic postage system 10. Fig. 8, shown below, is a block diagram of the console 13 shown in FIG. 1. As shown in Fig. 8, the console 13 includes a microprocessor 201 and memory (including ROM 213, memory section 250, flash EEPROM 250a, and EPROM 250b) that runs

an application program. Schwartz, 1:1-4, 6:27-8:67, Fig. 8, claim 1, 7:48-57 ("console 13 includes read-only-memory (ROM) 213 permanently storing a unique serial number pre-assigned to system 10. Console 13 also includes memory section 250 comprising flash electrically erasable programmable read-only-memory (EEPROM) 250a having a capacity of 128 Kbytes, EPROM 250b having a capacity of 32 Kbytes, a nonvolatile static random-access-memory (SRAM) 250c having a capacity of 128 Kbytes, and SRAM 250d having a capacity of 128 Kbytes, all of which are of conventional design."). A POSITA would have understood that Schwartz's device is a computer, because it has attributes of a computer such as a microprocessor, memory, display, and input/output components such as a keyboard. A POSITA would have further understood that EEPROM 250 is erasable and programmable, and is a non-volatile memory, because it is erased only if a desired signal is applied, and not by removing power to it.

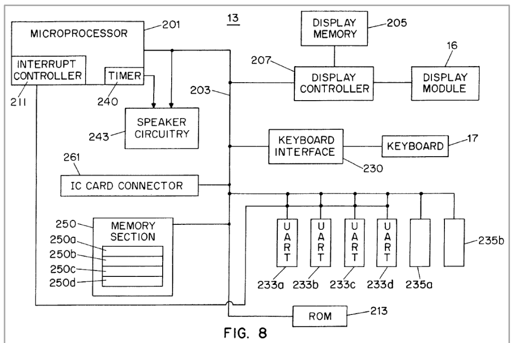


FIG. 8

574. A POSITA would have recognized Schwartz's SRAM 250d to be a common form of volatile memory, which also meets the Court's construction. See also Section XXV.

575. Moreover, the '941 patent does not limit its volatile memory claims (or specification) to any one form of RAM. '941 Patent, 5:15-16 ("a volatile memory area (6) (e.g., the internal RAM memory of the computer)"), 8:1-2 ("the volatile memory is a RAM").

576. Schwartz discloses that SRAM 250c is a non-volatile, static random-access-memory. Although Schwartz does not expressly state that SRAM 250d is a volatile memory, a POSITA would have understood that this is the case. For example, this is apparent from the

context in which SRAM 250c and SRAM 250d are mentioned together, with an express statement that SRAM 250c is non-volatile. This implies that SRAM 250d is volatile. More definitively, Schwartz states that SRAM 250d is used as "work space." Schwartz, 8:23-25. A POSITA would have understood that memory used by a computer-like device for "work space" would be a temporary volatile RAM memory. *See generally*, Silberschatz at 23–34 ("Computer System Structures"); *see also id.* at Figure 2.1 on p. 24, 31-32 ("Main memory and the registers built into the processor itself are the only storage that the CPU can access directly. (Consider that there are machine instructions that take memory addresses as arguments, but none that take disk addresses.) Therefore, any instructions in execution, and any data being used by the instructions, must be in one of these direct-access storage devices. If the data are not in memory, they must be moved there before the CPU can operate on them").

577. Fig. 9 of Schwartz (shown below) is a memory map of memory section 250 shown in Fig. 8. The memory section 250 includes: (1) flash EEPROM 250a, which is erasable and programmable non-volatile memory (Schwartz at 7:50-62), and (2) volatile memory, such as SRAM 250d. Schwartz, 7:54-56, 8:23-25. In Schwartz, the BIOS 309 contains a fixed portion (EPROM area 250b) and a programmable portion (EEPROM 250a). *Id.* at 8:14-20 ("Basic Input/Output System (BIOS) module 309 contains firmware responsible for basic machine operation of system 10. As shown in FIG. 9, zip/zone module 305, configuration module 307 and part of BIOS module 309 are located in flash EEPROM 250a. The

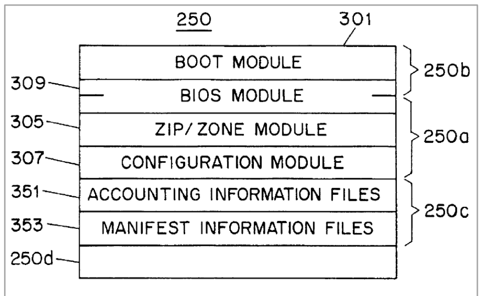


FIG. 9

remaining part of BIOS module 309 and boot module 301 are located in EPROM 250b."). A

POSITA would have understood that EEPROM 250a is "erasable, non-volatile memory area of a BIOS", given that it is programmable and is part of BIOS 309.

578. Although the Court ruled that this portion of the preamble is not limiting, Schwartz discloses restricting use of the system 10 and the running of the program, based on a verification process that involves an authorization number which is unique to the system 10. Schwartz, 11:21-40. The authorization number includes a 32-bit electronic signature and another 32-bit encrypted option segment. *Id.* at 11:21-40, 10:21-28 ("The authorization number, which is generated outside system 10 and provided to the user, is 64 bits long and consists of a 32-bit electronic signature and another 32-bit encrypted option segment"). The system 10 becomes operational only if the authorization number is verified. *Id.* at 11:21-40 ("After the system options are enabled in the prescribed manner, system 10 enters its normal mode of operation where it first validates the authorization number"). Schwartz's verification deters unauthorized copying of the software of system 10 onto other similar systems. *Id.* at 12:29-40 ("The authorization number verification requirement is desirable in that it helps deter unauthorized copying of software of system 10 onto other similar systems"). It thus "restricts software operation within a license for use with" Schwartz's device.

2. <u>Limitation 1[a] Is Disclosed or Rendered Obvious by Schwartz</u>

579. Limitation 1[a] of the '941 patent recites:

"selecting a program residing in the volatile memory;"

580. Schwartz discloses verifying an electronic signature that was previously stored in configuration module 307 with an electronic signature that is calculated based on the configuration of the system 10 and its software. Schwartz, 11:21-38. If there is a match, the application program is run, i.e., loaded into volatile memory such as SRAM 250d and executed

(which is a step of "selecting" from the volatile memory.) *Id*. at 11:38-40 ("Microprocessor 201 generates an electronic signature based on numbers (i) through (vi) using the aforementioned first encryption algorithm. The electronic signature, thus generated, is compared with the electronic signature stored in configuration module 307. If there is no mismatch, system 10 becomes operational. Otherwise if there is any mismatch, system 10 would prompt for a new authorization number"), 8:23-35 ("rate schedule data, an operating system and an application program (hereinafter referred to as the 'carrier service program') are provided to the user []. This application program when executed causes system 10 to perform certain tasks in accordance with the invention"). A POSITA would have understood that programs on conventional computers must be loaded into RAM (volatile memory) before being loaded into the CPU for execution (see Section IX.C above). Hence loading and executing a program, e.g., so it can be made operational, in Schwartz, is the act of "selecting" said program from volatile memory as per the Court's construction.

581. The application program is a computer program whose instructions are executed on system 10 (i.e., by microprocessor 201 of system 10). For that reason, it is a "program" under the Court's construction; Schwartz, 8:26-35, 8:63-65.

582. I understand that the Court has stated in its construction that the "selecting" step may happen at any time with respect to the other recited acts of claim 1. Thus, under the Court's construction, the "selecting" step may take place after other steps of claim 1, like setting up the verification structure, or verifying have been carried out.

3. ***Limitation 1[b] Is Disclosed or Rendered Obvious by Schwartz in Combination with Yee***

583. Limitation 1[b] of the '941 patent recites:

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

584. Schwartz in combination with Yee discloses or suggests this limitation.

a. <u>"Agent"</u>

585. Schwartz discloses verification software (the claimed "agent") (Schwartz, 8:26-35), running on system 10, that establishes at least one license-record location (as part of the claimed "set up verification structure"), and carries out verification based on the authorization number before rendering an application program of system 10 operational. Schwartz, 10:21-54 ("the user of system 10 needs to enter a valid authorization number, which is unique to system 10, in order to enable the new application software [...] The authorization number, which is generated outside system 10 and provided to the user, is 64 bits long and consists of a 32-bit electronic signature and another 32-bit encrypted option segment. [...] system 10 independently generates an electronic signature. The generated signature is compared with the electronic signature of the authorization number just entered. If the two signatures match, the authorization number is declared valid"), 11:24-40.

b. <u>"Verification Structure" and "License Record"</u>

586. Because verification in Schwartz is based on the electronic signature part of the authorization number, the authorization number corresponds to the "verification structure". Schwartz, 11:21-40.

587. The electronic signature of Schwartz corresponds to "license record". First, the electronic signature is part of the authorization number ("verification structure") which is used for verification. Schwartz, 10:21-54, 11:21-40. Second, the electronic signature is based on data that is associated with the application program, such as the version number of the application program, as well as option numbers that the application program can enable/disable

based on licensed features. *Id.* at 10:26-38 ("The authorization number, which is generated outside system 10 and provided to the user, is 64 bits long and consists of a 32-bit electronic signature and another 32-bit encrypted option segment. In order to generate the electronic signature, a combination of (a) the serial number of system 10, (b) the model number of system 10, (c) the version number of the application software, (d) the version number of the rate schedule data, (e) the version number of the zip/zone data, and (f) a 32-bit option number whose bit pattern corresponds to a particular combination of enabled and disabled system options, are first encrypted in accordance with a first encryption algorithm"). This satisfies the Court's construction's requirement that "license record" is "data associated with a licensed program with information for verifying that licensed program."

c. "Erasable, Non-volatile Memory of the BIOS"

588. I understand that the Court ruled that the terms "memory of the BIOS" and "non-volatile memory of the BIOS" need not be construed. I believe that a POSITA would have understood the term "memory of the BIOS" to require at least that it be a memory that stores the BIOS. I understand that the term "memory of the BIOS" has been construed by another court as having a plain and ordinary meaning, which that court ruled is "a memory that stores the BIOS." This construction also appears consistent with Ancora's application of the claim limitation in its Final Infringement Contentions. See, e.g., Ancora Final Infringement Contentions, Ex. B, p. 19, 27-29. Thus, I apply this construction here. To the extent that the Court adopts a different construction, I reserve the right to revise my opinion below.

589. As shown in Fig. 9 of Schwartz, configuration module 307 and part of BIOS module 309 are stored in the EEPROM 250a, which satisfies the claim element "erasable, non-volatile memory of the BIOS." First, EEPROM 250a stores configuration module 307, which stores part of the BIOS module. Schwartz, 8:16-20 ("As shown in FIG. 9, zip/zone module 305,

configuration module 307 and part of BIOS module 309 are located in flash EEPROM 250a.”). Second, EEPROM 205a is erasable, non-volatile memory. *Id*. at 7:50-57.

590. Schwartz also teaches that its authorization number (“verification structure”) and electronic signature (which is part of the authorization number and corresponds to “license record”) are stored (i.e., “set up”) in configuration module 307, and hence, EEPROM 250a. Schwartz, 10:50-54 (“the authorization number is declared valid; the authorization number will then be stored in a first memory buffer and the recovered option number will be stored in a second memory buffer in configuration module 307”), 11:36-37, 10:26-29.

591. Thus, Schwartz discloses that the authorization number (“verification structure”) and electronic signature (“license record”) are stored (“set up”) in EEPROM 250a (“erasable, non-volatile memory of the BIOS”). Additionally, storing a “license record” necessarily requires establishing a “license record location.”

d. <u>“Set Up a Verification Structure”, Including “Pseudo-Unique Key”</u>

592. Schwartz discloses that the system 10 calculates the electronic signature based on configuration of the system 10 using a first encryption algorithm. Schwartz, 10:25-38 (“The authorization number, which is generated outside system 10 and provided to the user, is 64 bits long and consists of a 32-bit electronic signature and another 32-bit encrypted option segment. In order to generate the electronic signature, a combination of (a) the serial number of system 10, (b) the model number of system 10, (c) the version number of the application software, (d) the version number of the rate schedule data, (e) the version number of the zip/zone data, and (f) a 32-bit option number whose bit pattern corresponds to a particular combination of enabled and disabled system options, are first encrypted in accordance with a first encryption algorithm”). The authorization number, which includes the electronic signature, is also calculated outside the system 10 and provided to the user of system 10. *Id*. at 10:21-28.

593. Schwartz discloses the use of at least one encryption algorithm. Schwartz, 10:21-42 ("In order to generate the electronic signature, a combination of [(a) through (f)] are first encrypted in accordance with a first encryption algorithm. The signature is then derived from the encrypted version of the combination of numbers (a) through (t). On the other hand, the encrypted option segment is generated by encrypting only the 32-bit option number in (f) in accordance with a second encryption algorithm"). Schwartz, however, does not expressly disclose the use of an encryption key in use with said encryption algorithms. A POSITA, however, would have understood that Schwartz discloses a highly secure system intended to deter software piracy. *Id*. at 12:29-40. Moreover, a POSITA would have known that for any encryption algorithm to be useful, practical, and secure enough, it must use keys to encrypt data. See Microsoft at 175 ("**encryption** [] *n.* The process of encoding data to prevent unauthorized access, especially during transmission. Encryption is usually based on a key that is essential for decoding. The U.S. National Bureau of Standards created a complex encryption standard, Data Encryption Standard (DES), which provides almost unlimited ways to encrypt documents. *See also* DES.), *Id.* ("**encryption key** [] *n.* A sequence of data that is used to encrypt other data and that, consequently, must be used for the data's decryption. *See also* decryption, encryption"). Conversely, simple, key-less encryption algorithms are not considered secure enough, for example the "ROT13" encryption method. See Microsoft at 414 ("**ROT13 encryption** [] *n.* A simple encryption method in which each letter is replaced with the letter of the alphabet 13 letters after the original letter, so that A is replaced by N, and so forth; N, in turn, is replaced by A, and Z is replaced by M. ROT13 encryption is not used to protect messages against unauthorized readers; rather, it is used in newsgroups to encode messages that a user may not want to read, such as sexual jokes or spoilers. Some newsreaders can automatically perform ROT13 encryption and

decryption at the touch of a key").

594. Schwartz does not disclose the details of its encryption algorithm. However, Yee discloses the use of a pseudo-unique key (which is required by the construction of the "set up verification structure" element) that is stored in erasable, non-volatile BIOS memory in Yee's Secure Processor shown in Fig. 3.1 (p. 21, seen below), and that can be used for symmetric-key encryption and decryption. Yee at 20-21, Fig. 3.1, 38, 1st and 2nd paragraphs. Yee's non-volatile memory is "BIOS" memory, because Yee discloses it may contain the bootstrap loader. Yee at 38, 1st paragraph. Thus, Yee establishes the existence of a pseudo-unique key" in non-volatile memory of the BIOS.

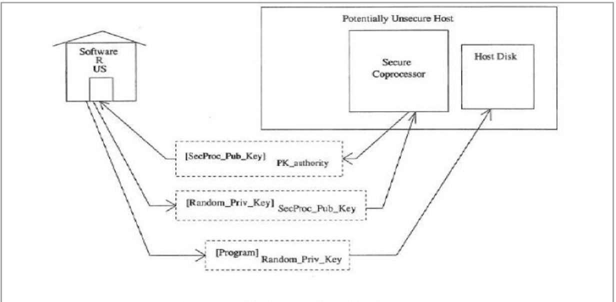

Potentially Unsecure Host
Software R US
Secure Coprocessor
Host Disk
[SecProc_Pub_Key] PK_authority
[Random_Priv_Key] SecProc_Pub_Key
[Program] Random_Priv_Key

Figure 3.1 Copy-Protected Software Distribution
The software retailer encrypts the copy-protected software with a random key. This key is encrypted using the public key of the secure coprocessor within the destination host, so only the secure coprocessor may decrypt and run the copy-protected software. The software retailer knows that the public key of the secure coprocessor is good, because it is digitally signed with the public key of the secure coprocessor distributor.

595. Yee's "Random_Priv_key" is pseudo-unique to the Yee Secure Processor shown in Fig. 3.1 (p. 21), because the Software R US shown in Fig. 3.1 assigns it to the Secure Processor for purposes of sending encrypted messages, which would not be intended to be read by third parties. Yee at 20 ("Secure coprocessors can protect executables from being copied and illegally used. The proprietary code to be protected—or at least some critical portion of it—is distributed and stored in encrypted form, so copying without the code decryption key is futile, and this protected code runs only inside the secure coprocessor"). If Software R US works with multiple client devices, then it would assign different, randomly generated software-decryption

keys to each of them, to ensure secure communications with each client device. This would enhance system security.

596. Further, "Random_Priv_key" is a symmetric key, because it is used by the server to encrypt messages sent to the client device, and by the client device to decrypt them. Yee at 20 ("Either public key or private key cryptography may be used to encrypt protected software"). A POSITA would recognize Yee referring to "private key cryptography" (as opposed to "public key cryptography"). See Section IX.D, above. See also Microsoft at 55 ("**block cipher** [] *n.* A private key encryption method that encrypts data in blocks of a fixed size (usually 64 bits). The encrypted data block contains the same number of bits as the original. *See also* encryption, private key"), 388–389 ("**public key encryption** [] *n.* An asymmetric scheme that uses a pair of keys for encryption: the public key encrypts data, and a corresponding secret key decrypts it. For digital signatures, the process is reversed: the sender uses the secret key to create a unique electronic number that can be read by anyone possessing the corresponding public key, which verifies that the message is truly from the sender. *See also* private *key,* public *key*").

597. The combination of Schwarz and Yee thus discloses the claimed "set up a verification structure" because Schwartz discloses setting up a verification structure, including a license-record location, and Yee discloses a pseudo-unique key that can be used, for example, in encryption that is carried out during Yee's system's execution of the "set up a verification structure" step.

4. <u>Limitation 1[c] Is Disclosed or Rendered Obvious by Schwartz</u>

598. Limitation 1[c] of the '941 patent recites:

> "verifying the program using at least the verification structure from the erasable non-volatile memory of the BIOS;"

599. Schwartz discloses verifying the application program by comparing the electronic

signature that is stored in configuration module 307 (which is part of EEPROM 250a, the "memory of the BIOS") with an electronic signature that is calculated based on the configuration of the system 10 and its software. Schwartz, 11:21-40 ("After the system options are enabled in the prescribed manner, system 10 enters its normal mode of operation where it first validates the authorization number. In the normal mode of operation or each time when system 10 is powered up, microprocessor 201 reads off (i) the serial number of system 10 from ROM 213, (ii) the model number of system 10 which is stored in BIOS module 309, (iii) the version number of the application software which is stored in the application module, (iv) the version number of the rate schedule data which is stored in the rate module, (v) the version number of the zip/zone data which is stored in module 305, and (vi) the option number which is stored in configuration module 307. Microprocessor 201 generates an electronic signature based on numbers (i) through (vi) using the aforementioned first encryption algorithm. The electronic signature, thus generated, is compared with the electronic signature stored in configuration module 307. If there is no mismatch, system 10 becomes operational. Otherwise if there is any mismatch, system 10 would prompt for a new authorization number"). Thus, the application program is verified by using the authorization number ("verification structure"), which includes the electronic signature ("license record"). *Id.*

5. __Limitation 1[d] Is Disclosed or Rendered Obvious by Schwartz__

600. Limitation 1[d] of the '941 patent recites:

"acting on the program according to the verification."

601. Schwartz discloses or suggests this limitation. If there is a match in the verification step, the application program is run. Schwartz at 11:38-40 ("The electronic signature, thus generated, is compared with the electronic signature stored in configuration module 307. If there is no mismatch, system 10 becomes operational. Otherwise if there is any

mismatch, system 10 would prompt for a new authorization number"), 8:26-35. If not, the program is not run.

B. Claim 2 Is Rendered Obvious by the Combination of Schwartz and Yee

602. Claim 2 of the '941 patent recites:

> A method according to claim 1, further comprising the steps of: establishing a license authentication bureau.

603. Claim depends from claim 1, which is obvious over the combination of Schwartz and Yee as discuss above. Moreover, Schwartz discloses or suggests the above additional limitation of claim 2. As discussed above in claim 1, Schwartz discloses that system 10 generates and stores the electronic signature. Schwartz, 10:25-54. It also discloses verifying the application program by comparing the stored electronic signature with one generated on the fly. *Id*. at 11:21-38. If the comparison is successful, the application program is run. *Id*. at 11:38-40, 8:2635. If not, the application program is not run.

604. The '941 patent does not expressly define "license authentication bureau," explaining that it may either be an "external entity" or "may form part of the computer." '941 Patent at 6:1-3. Its functions may include "verifying if the key and license record in the non-volatile memory(s) is compatible with the license record information as extracted from the application under question." *Id*. at 3:42-44, 5:50-52. Further, the bureau may "form[] the proposed license-record from the contents" of the program. *Id*. Because Schwartz discloses that system 10 both generates the electronic signature and uses it for verification, system 10 is a "license authentication bureau."

605. Schwartz also discloses an embodiment in which data, such as the new versions of the application program, are obtained over a network instead of from an IC card. Schwartz, Fig. 33 (right), 21:40-64, 9:33-21. In this embodiment, personal computer 187 would provide the authorization number to system 10 and the user, instead of the IC card. *Id.* at 10:15-20, 10:25-27, 21:40-42, Fig. 33. Thus, personal computer 187, instead of the IC card, would form the electronic signature (i.e., the "license record") and provide it to the system 10. *Id.* at 10:25-42, 11:40-42. In this embodiment, personal computer 187 is a "license authentication bureau." To the extent Schwartz does not disclose a license authentication bureau, the combination of Schwartz and Yee does, as detailed in Section XXVI.M.4 below.

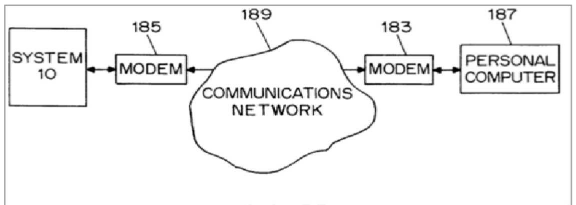


FIG. 33

C. Claim 6 Is Rendered Obvious by the Combination of Schwartz and Yee

606. Claim 6 of the '941 Patent recites:

> 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.

607. Claim 6 depends from claim 1, which is obvious over the combination of Schwartz and Yee as discuss above. Moreover, Schwartz discloses the above additional limitation of claim 6.

608. As discussed above in connection with the "selecting" step of claim 1, Schwartz discloses establishing the software program in SRAM 250d ("volatile memory") of the console 13 ("the computer"). Schwartz, 8:23-25, 8:63-65. The application program software includes contents in the form of a version number that is also used to generate the electronic signature

("license record"). *Id*. at 11:40-42, 11:24-36, 10:26-38. The electronic signature is used to determine if an authorized version of the software program is installed on the system 10, in which case the system becomes operational. *Id*. at 11:36-40. See also Section IX.G, above.

D. Claim 7 Is Rendered Obvious by the Combination of Schwartz and Yee

1. ***Limitation 7[a] Is Rendered Obvious by the Combination of Schwartz and Yee***

609. Limitation 7[a] of the '941 patent recites:

> A method according to claim 6 wherein using an agent to set up the verification structure includes the steps of: establishing or certifying the existence of a pseudo-unique key in a first non- volatile memory area of the computer;

610. *See* discussion of Limitation 1[b], element (d), above.

2. ***Limitation 7[b] Is Rendered Obvious by the Combination of Schwartz and Yee***

611. Limitation 7[b] of the '941 patent recites:

> establishing at least one license-record location in the first nonvolatile memory area or in the erasable, nonvolatile memory area of the BIOS.

612. *See* discussion of Limitation 1[b] element (d), above.

E. Claim 8 Is Rendered Obvious by the Combination of Schwartz and Yee

1. ***Limitation 8[a] Is Rendered Obvious by the Combination of Schwartz and Yee***

613. Limitation 8[a] of the '941 patent recites:

> A method according to claim 6 wherein establishing a license- record includes the steps of: forming a license-record by encrypting of the contents used to form a license-record with other predetermined data contents, using the key

614. Schwartz discloses that the version number of its application program is part of the program. Schwartz, 11:41-43. The electronic signature ("license record") is generated based on encrypting the version number and other configuration-related data of the device. *Id*. at

11:24-36, 10:26-38. *See also* discussion of Limitation 1[b], element (d), above.

615. As discussed above in connection with Limitation 1[b], element (d), above, a POSITA would have understood that Yee's method of encryption using a symmetric key assigned to the device can be used for generating Schwartz's electronic signature. Yee at 20-21, Fig. 3.1. Because Schwartz discloses that the electronic signature is generated based on encryption of, among others, the version number, the combination of Schwartz and Yee discloses this element.

2. Limitation 8[b] Is Rendered Obvious by the Combination of Schwartz and Yee

616. Limitation 8[b] of the '941 patent recites:

> establishing the encrypted license-record in one of the at least one established license-record locations.

617. See discussion of Limitation 1[b], element (b), above.

F. Claim 9 Is Rendered Obvious by the Combination of Schwartz and Yee

1. Limitation 9[a] Is Disclosed by the Combination of Schwartz and Yee

618. Limitation 9[a] of the '941 patent recites:

> A method according to claim 7 wherein verifying the program includes the steps of: encrypting the licensed-software-program's license-record contents from the volatile memory area or decrypting the license-record in the erasable, nonvolatile memory area of the BIOS, using the pseudo-unique key;

619. Schwartz in combination with Yee discloses "encrypting the licensed-software-program's license-record contents from the volatile memory area." Regarding the electronic signature discussed above with respect to Limitation 1[c] in connection with the verification step, Schwartz discloses that system 10 calculates the electronic signature based on encrypting configuration data including the version number of the application program. Schwartz, 11:21-38. Schwartz, however, does not disclose the details of the encryption algorithm.

620. As discussed above in connection with Limitation 1[b], element (d), system 10 may use a symmetric, pseudo-unique key that is assigned to it to carry out the encryption step for generating the electronic signature, based on the disclosure of Yee. Yee at 20-21, Fig. 3.1. As discussed above in connection with the "selecting" step of claim 1 (i.e., Limitation 1[a]), such processing by the microprocessor 201 of the system 10 would be carried out by writing data used for such processing into volatile memory (e.g., SRAM 250d) used by the microprocessor 201. Schwartz, 7:54-56, 8:23-25.

2. <u>Limitation 9[b] Is Disclosed by the Combination of Schwartz and Yee</u>

621. Limitation 9[b] of the '941 patent recites:

> comparing the encrypted licenses-software-program's license- record contents with the encrypted license-record in the erasable, nonvolatile 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.

622. Schwartz discloses comparing the (encrypted) electronic signature that is stored in configuration module 307 ("the encrypted licenses-software-program's license-record contents") with the calculated electronic signature ("the encrypted license-record in the erasable, non-volatile memory area of the BIOS"). Schwartz, 11:21-40. Configuration module 307 is part of the erasable and programmable non-volatile, flash EEPROM 250a (which corresponds to the recited "erasable, non-volatile memory of the BIOS," as discussed above in connection with Limitation 1[b], element (c).

G. Claim 10 Is Rendered Obvious by the Combination of Schwartz and Yee

623. Claim 10 of the '941 patent recites:

> 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.

624. Claim 10 depends from claim 9, which is obvious over the combination of

Schwartz and Yee as discuss above. Moreover, Schwartz discloses or suggests the above additional limitation of claim 10.

625. According to Schwartz, if there is no match in the verification step, the application program is not run ("restricting the program's operation with predetermined limitations"). Schwartz, 11:38-40. See also discussion of Limitation 1[d] above.

H. Claim 11 Is Rendered Obvious by the Combination of Schwartz and Yee

626. Claim 11 of the '941 patent recites:

A method according to claim 1 wherein the volatile memory is a RAM.

627. Claim 11 depends from claim 1, which is obvious over the combination of Schwartz and Yee as discuss above. Moreover, Schwartz and Yee disclose or suggest the above additional limitation of claim 11.

628. Each of Schwartz and Yee discloses or suggests this limitation. As discussed in connection with the preamble of claim 1 (i.e., Limitation 1[preamble]), above, Schwartz discloses SRAM 250d, which is RAM volatile memory. Schwartz, 11:38-40, 8:26-35.

629. Yee also discloses RAM volatile memory. Yee at 39 ("The PS/2 host contains an Intel i386 CPU running at 16 MHz, 16 megabytes of RAM, and a microchannel system bus. The Citadel coprocessor contains an Intel i386SX CPU, also running at 16 MHz; one megabyte of 'scratch' volatile RAM; 64 kilobytes of battery-backed secure RAM; bootstrap EPROM with a simple monitor program; and 64 kilobytes of second-stage bootstrap EEPROM; an IBM produced DES chip with a theoretical throughput of 30 megabytes per second"). A POSITA would have understood that RAM volatile memory would be used by a processor in the console 13 during processing. See Section IX.

I. Claim 12 Is Rendered Obvious by the Combination of Schwartz and Yee

630. Claim 12 of the '941 patent recites:

The method of claim 1, wherein a pseudo-unique key is stored in the non-volatile memory of the BIOS.

631. Claim 12 depends from claim 1, which is obvious over the combination of Schwartz and Yee as discuss above. Moreover, Yee discloses or suggests the above additional limitation of claim 12.

632. Yee discloses or suggests this limitation. As discussed above in connection Limitation 1[b], element (d), a POSITA would have understood that symmetric key cryptography (also known as shared or private key cryptography) involves two parties using the same private key to both encrypt and decrypt information. Yee discloses that the "private key is stored only within the secure nonvolatile memory of the secure coprocessor." Yee at 20, Section 3.3.1, first paragraph. A POSITA would have further understood that "the secure nonvolatile memory of the secure coprocessor" would be located in and be a part of the BIOS memory of the secure coprocessor. *Id.* at 20-21, Fig. 3.1, 38 (1st and 2nd paragraphs), 39 ("The PS/2 host contains an Intel i386 CPU running at 16 MHz, 16 megabytes of RAM, and a microchannel system bus. The Citadel coprocessor contains an Intel i386SX CPU, also running at 16 MHz; one megabyte of 'scratch' volatile RAM; 64 kilobytes of battery-backed secure RAM; bootstrap EPROM with a simple monitor program; and 64 kilobytes of second-stage bootstrap EEPROM; an IBM produced DES chip with a theoretical throughput of 30 megabytes per second"). Based on these disclosures, a POSITA would have understood that the pseudo-unique key in the Schwartz system as modified by Yee's teaching would be stored in non-volatile memory of the BIOS, i.e., EEPROM 250a of Schwartz.

J. Claim 13 Is Rendered Obvious by the Combination of Schwartz and Yee

633. Claim 13 of the '941 patent recites:

The method of claim 1, wherein a unique key is stored in a first non-volatile memory area of the computer.

634. Claim 13 depends from claim 1, which is obvious over the combination of Schwartz and Yee as discuss above. Moreover, the combination of Schwartz and Yee discloses or suggests the above additional limitation of claim 13.

635. As discussed above in Limitation 1[b], element (d), Yee discloses storing a “Random_Priv_Key” (e.g., a symmetric key assigned to and used by a client device to decrypt an encrypted message from a server, as discussed with Fig. 3.1 of Yee) in non-volatile memory that also stores the BIOS. Yee at 20-21, Fig. 3.1, 38, 1st and 2nd paragraphs. Furthermore, Schwartz discloses that its authorization number, which includes the electronic signature that is derived using encryption, is “unique to system 10”. Schwartz, 10:21-25. A POSITA implementing Yee’s private key for encryption in Schwartz would have understood that to ensure uniqueness of the electronic signature and the authorization number, she would have to pick “Random_Priv_Key” such that it is a unique key. See also Section IX.E and IX.F, above. This would strengthen the security of the system disclosed in Schwartz, and ensure that the authorization number is unique as specified by Schwartz.

K. Claim 14 Is Rendered Obvious by the Combination of Schwartz and Yee

636. Claim 14 of the ’941 patent recites:

> 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.

637. *See* discussion of claims 8 and 13, above.

L. Claim 16 Is Rendered Obvious by the Combination of Schwartz and Yee

638. Claim 16 of the ’941 patent recites:

> 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.

639. Claim 16 depends from claim 13, which is obvious over the combination of Schwartz and Yee as discuss above. Moreover, Schwartz discloses or suggests the above additional limitation of claim 16.

640. Schwartz discloses an alternate verification method in which the device decrypts the encrypted authorization number, and compares the result with an entered authorization. Schwartz, 12:15-29 ("In accordance with this alternative technique, the authorization number is generated by encrypting the above numbers (a) through (f) using a standard encryption algorithm. After the user enters such an authorization number, system 10 decrypts the entered number using a decryption algorithm inverse to the standard encryption algorithm, and recovers the underlying numbers (a) through (f). System 10 then retrieves the above numbers (i) through (v) in the manner described before, and compares them with the corresponding, recovered numbers (a) through (e). The authorization number is validated if the two sets of numbers match"). Based on this disclosure, a POSITA would have understood that the verification step of claim 1 could be carried out by decryption of an encrypted license record, rather than encryption of the license record.

M. Motivation to Combine Schwartz and Yee and/or POSITA knowledge

641. In my opinion, it would have been obvious to a POSITA to modify the Schwartz secure postage system based on the teachings of Yee and/or knowledge of POSITA.

642. I provide below rationales for why this combination would have been obvious and successful.

1. Schwartz and Yee are in the Same Field

643. Schwartz and Yee are directed to the same field of secure postage systems that require careful verification and authorization of software. Just as software copyright violations and piracy is a serious offense, stealing or forging postage stamps is a serious crime. A POSITA

would have known that postage is tantamount to cash as it carries monetary value, so methods of circumventing secure postage systems can lead to monetary theft.

644. Schwartz expressed such concerns. Schwartz, 2:45-48 ("Another object of the invention is to prevent or deter unauthorized copying of software provided for a postage scale system, and to easily enable selected system options using an authorization number.").

645. Even back in 1994, Yee recognized this growing problem. Yee at 28 ("Postal meters are subject to at least four types of attack: (1) the postage meter recorded credit may be tampered with, allowing the user to steal postage; (2) the postage meter stamp may be forged or copied; (3) a valid postage meter may be used by an unauthorized person; and (4) a postage meter may be stolen."), *id.* ("82,000 franking machines in the U.S. are currently reported as lost or stolen [85]"), 19 ("Secure coprocessors can protect executables from being copied and illegally used. The proprietary code to be protected - or at least some critical portion of it - is distributed and stored in encrypted form, so copying without the code decryption key is futile").

646. Both Schwartz and Yee share the common goal of ensuring security for electronic postage systems. This also includes the goal of providing security against piracy of the program running on these systems. Schwartz, 2:45-48 ("Another object of the invention is to prevent or deter unauthorized copying of software provided for a postage scale system, and to easily enable selected system options using an authorization number."); Yee at 20-21, 31-33, 28-33 ("Secure Postage" section). Therefore, a POSITA considering Schwartz would have easily found Yee's teaching when investigating the general problem area of secure postage systems.

2. Schwartz and Yee Use Similar Techniques

647. Both Schwartz and Yee use similar techniques. For example, they use signatures as software identifiers and for integrity, as well as encryption to protect sensitive information.

648. Schwartz, 10:30-39 ("to generate the electronic signature, a combination of (a)

[...] and (f) [...] are first encrypted in accordance with a first encryption algorithm. The signature is then derived from the encrypted version of the combination of numbers (a) through (f)."), 10:43-51 ("after the user enters the authorization number, its encrypted option segment is first decrypted to recover the underlying option number. With the recovered option number, and additional numbers, system 10 independently generates an electronic signature. The generated signature is compared with the electronic signature of the authorization number just entered. If the two signatures match, the authorization number is declared valid").

649. Yee at 20 ("Either public key or private key cryptography may be used to encrypt protected software. [...] when a user pays for the use of a program, he sends the certificate of his secure coprocessor public key to the software vendor. This certificate is digitally signed by a key management center and is evidence that the public key is valid. The corresponding private key is stored only within the secure non-volatile memory of the secure coprocessor; thus, only the secure coprocessor will have full access to the proprietary software."), 29 ("A cryptographic postage stamp is an indicia that can demonstrate to the postal authorities that postage has been paid. [...] We use cryptography to provide a crucial property: the stamp depends on the address. [...] To achieve this goal, we encrypt (or cryptographically checksum) as part of the stamp information relevant to the delivery of the particular piece of mail - e.g., the return address and the destination address, the postage amount, and class of mail, etc, as well as other identifying information, such as the serial number of the postage meter, a serial number for the stamp, and the date/time (a timestamp). The information, including the cryptographic signature or checksum, is put into a barcode.").

650. Therefore, a POSITA reviewing Schwartz's techniques (encryption, digital or electronic signatures) would have likely also identified Yee (and other references). A POSITA

would have also known that such techniques were well known in the art. See Microsoft at 388–389 ("**public key encryption**").

3. Schwartz and Yee are From the Same Time Frame

651. The 1990s was an unprecedented period where the Internet and Web came to be, with many traditional (e.g., paper) technologies moving to computerized and Internet-connected forms—also exposing new security risks that had to be mitigated. A POSITA would have been even more likely to find both Schwartz and Yee because they are from the same time frame. Schwartz was filed in June 1995, claiming priority to an earlier U.S. patent application filed in October 1993. Yee was published in May 1994 by Carnegie Mellow University (CMU).

4. Motivation to Improve on Schwartz's Network Server System with Yee's Remote Network Software Repository

652. Schwartz discloses operations such as encryption and electronic signatures that are executed "outside system 10" (the client). Schwartz, 10:26-29 ("The authorization number, which is generated outside system 10 and provided to the user, is 64 bits long and consists of a 32-bit electronic signature and another 32-bit encrypted option segment.").

653. Schwartz details what information is used and how to produce authorization numbers. Schwartz, 10:29-42 ("to generate the electronic signature, a combination of (a) the serial number of system 10, (b) the model number of system 10, (c) the version number of the application software, (d) the version number of the rate schedule data, (e) the version number of the zip/zone data, and (f) a 32-bit option number whose bit pattern corresponds to a particular combination of enabled and disabled system options, are first encrypted in accordance with a first encryption algorithm. The signature is then derived from the encrypted version of the combination of numbers (a) through (f).").

654. In the above two disclosures, Schwartz uses passive voice language and does not

directly state that an external server is performing these actions.

655. Nevertheless, Schwartz discloses an embodiment, illustrated in FIG. 33 (right), where the client (system 10) clearly communicates with a separate system over a network. Schwartz, 21:40-64 ("data is transported from system 10 to a similar system using an IC card. [...] download data such as accounting files from system 10 through a communications network to a remote computer which, for example, handles the billing. To this end, FIG. 33 illustrates an exemplary arrangement where personal computer 187 of conventional design is configured to communicate with system 10 via standard modem 183. [...] computer 187 is equipped with commercial communication software [...] configured to communicate via modem 185 which is compatible with modem 183. With this arrangement, system 10 is capable of transferring its accounting files to computer 187 over communications network 189 [...] Moreover, with this arrangement, computer 187 can be used, in lieu of the IC cards, to download through communications network 189 data such as the rate schedule data and the zip/zone data, and the carrier service application program to system 10 for updating the system."). Here, Schwartz describes details of how system 10 could communicate with a separate server bi-directionally, and what information would be exchanged between the two.

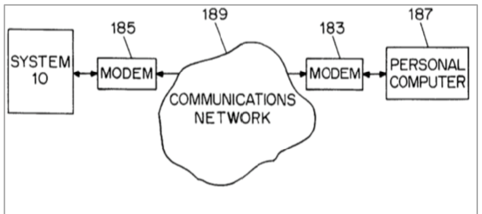


FIG. 33

656. A POSITA reading the above passages would have understood that clearly: said server outside system 10 ***is*** the one generating the electronic signature and encrypted option segment.

657. To the extent a POSITA reading Schwartz alone would not find clear disclosures

of the recited "bureau," Yee discloses that.

658. Yee clearly discloses a software clearing house, named "Software R US", where license information is generated in encrypted form. Yee at 21 ("Figure 3.1: Copy-Protected Software Distribution. The software retailer encrypts the copy-protected software with a random key. This key is encrypted using the public key of the secure coprocessor within the destination host, so only the secure coprocessor may decrypt and run the copy-protected software. The software retailer knows that the public key of the secure coprocessor is good, because it is digitally signed with the public key of the secure coprocessor distributor."), Figure 3.1 (reproduced below, demonstrating bi-directional communications).

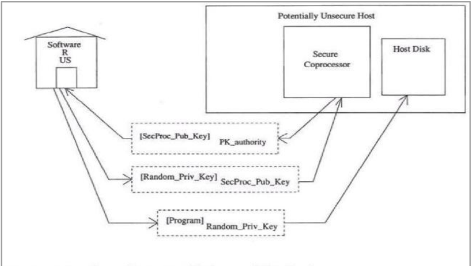

Potentially Unsecure Host
Software R US
Secure Coprocessor
Host Disk
[SecProc_Pub_Key] PK_authority
[Random_Priv_Key] SecProc_Pub_Key
[Program] Random_Priv_Key

Figure 3.1 Copy-Protected Software Distribution

659. Yee further details these server-side operations to include encryption, users paying for software, sending their requests for license and identification, and receiving from the server a digital certificate that only the legitimate users can use. Yee at 20 ("The proprietary code to be protected - or at least some critical portion of it - is distributed and stored in encrypted form, so copying without the code decryption key is futile [...] Either public key or private key cryptography may be used to encrypt protected software. [...] In particular, when a user pays for the use of a program, he sends the certificate of his secure coprocessor public key to the software vendor. This certificate is digitally signed by a key management center and is *prima facie* evidence that the public key is valid. [...] Figure 3.1 diagrams this process.").

660. A POSITA would have been motivated to modify Schwartz's external server to include Yee's public/private key cryptography because these techniques provide greater

assurance of the identity of both the client that requested the license, and the server that issued it.

661. Both private and public key cryptography, and digital signatures, were well known techniques in the art. A POSITA would have recognized that using them in Schwartz is a simple application of known techniques. See Microsoft at 55, 388–389.

662. Moreover, a POSITA would have had a reasonable expectation of success. Schwartz discloses that it uses a "standard encryption algorithm" for *both* encryption and decryption. Schwartz, 12:17-22 ("the authorization number is generated by encrypting the above numbers (a) through (f) using a standard encryption algorithm. After the user enters such an authorization number, system 10 decrypts the entered number using a decryption algorithm inverse to the standard encryption algorithm"). A POSITA would have recognized that the cryptographic techniques disclosed in Yee were well known in the art since the 1980s and hence one could replace whatever encryption algorithm Schwartz contemplates with another with relative ease.

5. Motivation to Improve on Schwartz's Use of Keys and Ciphers with Yee's Disclosures

663. Schwartz discloses using encryption, digital signatures, and using a standard encryption algorithm. Schwartz, 10:29-42 ("to generate the electronic signature, a combination of (a) the serial number of system 10, (b) the model number of system 10, (c) the version number of the application software, (d) the version number of the rate schedule data, (e) the version number of the zip/zone data, and (f) a 32-bit option number whose bit pattern corresponds to a particular combination of enabled and disabled system options, are first encrypted in accordance with a first encryption algorithm. The signature is then derived from the encrypted version of the combination of numbers (a) through (f)."); 12:17-22 ("the authorization number is generated by encrypting the above numbers (a) through (f) using a standard encryption algorithm. After the

user enters such an authorization number, system 10 decrypts the entered number using a decryption algorithm inverse to the standard encryption algorithm").

664. Nevertheless, Schwartz does not directly disclose (1) that it uses an encryption key and (2) what it uses as the encryption key.

665. First, a POSITA would have recognized that *any* useful encryption algorithms (i.e., a cipher), inherently requires at least one encryption key (e.g., symmetric ciphers), and may even use two keys (a "public" and "private" key pair used in public-key cryptography). To the extent a that Schwartz is interpreted to omit the use of at least one key for encryption, Yee discloses it. See discussion of Limitation 1[b], element (d), above.

666. Second, Schwartz discloses what information is generally used to generate authorization number, which includes an electronic signature and encryption option string. Schwartz, 10:29-42.

667. A POSITA would have been motivated to use Yee's pseudo-unique "Random_Priv_key" with the system 10 of Schwartz because Schwartz does not expressly disclose the details of its encryption algorithm. See discussion of Limitation 1[b], element (d), above. As disclosed in Yee, the symmetric key can be generated outside the device and provided to the device. Yee at 20-21, Fig. 3.1. Moreover, when combined with Schwartz, Yee's symmetric key can be used to calculate the authorization number both outside Schwartz's system 10 and within Schwartz's system 10, consistent with Schwartz's disclosure. Schwartz, 10:25-27, 10:45-54.

668. Combining Yee's key in this manner into Schwartz's disclosure is just applying a known technique (encryption using a key) to a known device (the system 10 of Schwartz) that yields a predictable result—a more secure electronic postage system. Because both Schwartz and

Yee disclose electronic postage systems and methods for providing security for such systems, a POSITA would have been motivated to adopt a known technique in one into the device of the other. Schwartz, 1:1-4, claim 1; Yee at 28-33.

669. Yee discloses an additional security-related feature—the use of a secure co-processor. A POSITA would have known that a secure co-processor offers very high levels of security, because it is designed to be a tamper-proof device. See Yee at 7 ("To prevent direct intrusion, these systems incorporate sensors consisting of fine (40 gauge) nichrome wire and low power sensing circuits powered by a long-lived battery. [...] the entire assembly is then dipped in epoxy. [...] The sensing electronics detect open circuits or short circuits in the wires and erase non-volatile memory if intrusion is attempted. Physical intrusion by mechanical means (e.g., drilling) cannot penetrate the epoxy without breaking one of these wires."). A POSITA would have recognized these high tamper proof properties.

670. A POSITA might choose to implement the secure postage system of Schwartz using a secure co-processor, based on Yee's teaching of a secure coprocessor. This would entail additional effort, time, and expense on the part of the implementer, in part because secure co-processors are added cost to a computer system, but it would lead to an even more secure device. However, an implementer who wanted to ensure strong security without undertaking additional effort and expense would, for example, not implement Yee's physical secure coprocessor, but rather follow the well-known techniques outlined in Yee. For example, implementing the Schwartz device using Yee's encryption techniques would be easy, straightforward, fairly secure, and would not incur additional cost: a person would, for example, use Yee's symmetric software encryption keys (transmitted securely in a previous message authenticated using asymmetric encryption) to ensure the authenticity and authority of the software vendor. A POSITA would

have recognized that Yee's techniques as outlined in Figure 3.1 could be accomplished entirely in software and without an additional co-processor. See Yee at 21, Fig. 3.1. See also Sections IX.E and IX.F.

671. Another motivation to combine Schwartz and Yee is their shared purpose for using cryptography—to authenticate both software vendor and client/user who uses licensed software.

672. See Schwartz at Fig. 12, 2:45-48 ("Another object of the invention is to prevent or deter unauthorized copying of software provided for a postage scale system, and to easily enable selected system options using an authorization number"), 10:21-54 ("In accordance with still another aspect of the invention, the user of system 10 needs to enter a valid authorization number, which is unique to system 10, in order to enable the new application software, or other new data or system options selected by the user. The authorization number, which is generated outside system 10 and provided to the user, is 64 bits long and consists of a 32-bit electronic signature and another 32-bit encrypted option segment. In order to generate the electronic signature, a combination of [...] are first encrypted in accordance with a first encryption algorithm. The signature is then derived from the encrypted version of the combination of numbers (a) through (f). [...] It suffices to know for now that after the user enters the authorization number, its encrypted option segment is first decrypted to recover the underlying option number. With the recovered option number, and additional numbers, system 10 independently generates an electronic signature. The generated signature is compared with the electronic signature of the authorization number just entered. If the two signatures match, the authorization number is declared valid").

673. Similarly, Yee discloses strong authentication mechanisms to ensure software is

distributed legitimately. Yee at 20 ("when a user pays for the use of a program, he sends the certificate of his secure coprocessor public key to the software vendor. This certificate is digitally signed by a key management center and is *prima facie* evidence that the public key is valid. The corresponding private key is stored only within the secure non-volatile memory of the secure coprocessor; thus, only the secure coprocessor will have full access to the proprietary software. Figure 3.1 diagrams this process."), Fig. 3.1 caption ("The software retailer encrypts the copy-protected software with a random key [Random_Priv_Key]. This key is encrypted using the public key of the secure coprocessor within the destination host, so only the secure coprocessor may decrypt and run the copy-protected software. The software retailer knows that the public key of the secure coprocessor is good, because it is digitally signed with the public key of the secure coprocessor distributor."), 28 ("Postal meters are subject to at least four types of attack: (1) the postage meter recorded credit may be tampered with, allowing the user to steal postage; (2) the postage meter stamp may be forged or copied; (3) a valid postage meter may be used by an unauthorized person; and (4) a postage meter may be stolen."), 28-29 ("These indicia could encode a digitally signed message which would guarantee authenticity. If this digital information included unique data about the letter (such as the date mailed, zip codes of the originator and recipient, etc.), the digitally signed stamp could protect against forged or copied stamps."). Additionally, when the host receives an encrypted message containing encrypted software program, it will attempt to decrypt it with Random_Priv_Key, and then execute the program. If this process succeeds and the program executes successfully, then the host can be sure that the server sent the message. A POSITA would have known that if the program would be decrypted with the wrong Random_Priv_Key, it would result in a bad executable that could not run. This is because only the server knows that host's

Random_Priv_Key and the entire message was authenticated using the server's public/private keys.

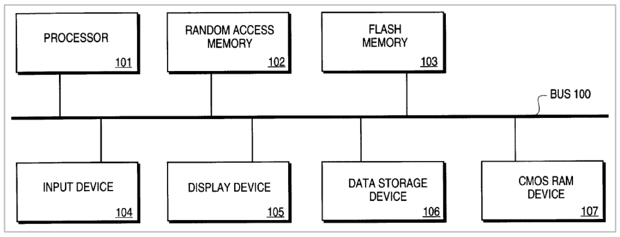
674. A POSITA comparing the above disclosures in Schwartz and Yee would have recognized that both of them use similar encryption techniques to protect licensed software and licenses, as well as ensure the authenticity and authorization of the software distributed and the vendor that distributes it. Moreover, the techniques of encryption used, such as Yee's symmetric encryption using a private key, were well established before the '941 patent was filed. For that reason, a POSITA would expect to succeed in using an encryption method like Yee's in the encryption method of the postal meter device of Schwartz.

XXVII. OVERVIEW OF THE CHRISTESON PATENT

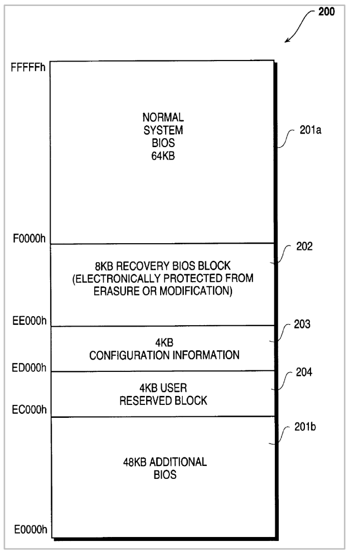
675. Christeson was filed on September 29, 1995 and issued on October 13, 1998. The ’941 Patent was filed on October 1, 1998, claiming priority to an Israeli patent application filed on May 21, 1998. Therefore, I understand that Christeson qualifies as prior art under Pre-AIA 35 U.S.C. § 102(e) because it is a patent that issued from a patent application filed in the U.S. before May 21, 1998 or the earliest conception date asserted by Ancora for the ’941 Patent.

676. Christeson discloses a computer system in which a FLASH memory is used to back up the CMOS RAM configuration and to store system software like BIOS. Christeson, Figures 1 and 2, 3:64-4:61.

677. Figure 1, reproduced below, illustrates a computer system of Christeson, including FLASH memory 103:



678. Figure 2 (reproduced below) illustrates the FLASH memory device of the preferred embodiment of Christeson storing BIOS:



XXVIII. OVERVIEW OF THE HELLMAN PATENT

679. Hellman was filed on July 11, 1983 and issued on April 14, 1987. The ’941 Patent was filed on October 1, 1998, claiming priority to an Israeli patent application filed on May 21, 1998. Regardless of the priority date or the earliest invention date claimed by Ancora, because Hellman issued more than one year before the earliest filing date of the ’941 Patent in the United States (i.e., October 1, 1998), I understand that Hellman qualifies as prior art to the ’941 Patent on the basis of pre-AIA 35 U.S.C. § 102(b).

680. Hellman 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, Abstract. The authorization for additional uses comes from the software’s manufacturer. *Id.* Hellman discloses a technique whereby the authorization message cannot be reused. *Id.* The authorization can be specific to a base unit, “so that an authorization for one base unit cannot be transferred to another base unit.” *Id.* Hellman indicates that its

technique "solves the 'software piracy problem'." *Id.* As such, Hellman seeks to solve the same problem as the '941 Patent.

681. 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." Hellman, 1:15-16.

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

683. 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. Hellman, 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. *Id.* at 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." *Id.* at 4:34-36.

684. 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. Hellman, 5:39-50. The user of base unit 12 obtains "software package

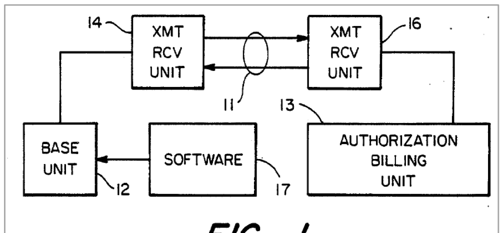


FIG_1

17 by purchasing it at a store, over telephone line, or in some similar manner." *Id.* at 5:51-56.

685. The base unit 12 generates a "user originated request for software use" for the

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

686. The base unit 12 transmits the request to the authorization and billing unit 13 over the insecure channel 11. Hellman, 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. *Id*. at 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. *Id*. at 6:3-15.

687. 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. Hellman, 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. *Id*. at 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

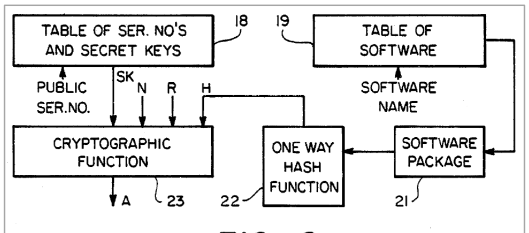


FIG_2

software package 21 is identical to software package 17. *Id*. at 6:16-30.

688. A one-way hash function 22 take the software package 21 as input and generates a hash value H. *Id*. at 6:31-61. "This output H is used as an 'abbreviation' or name for describing the software package 21." *Id*. at 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." *Id*. at 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." *Id*. at 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. *Id*. at 6:31-61.

689. 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. Hellman, 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. *Id*. at 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). *Id*. at 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). *Id*. at 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. *Id*. at 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. *Id*. at 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.

690. Hellman depicts the operation of the base unit 12 during verification of authorization A in Figure 6 (right). 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. Hellman, 9:16-28.

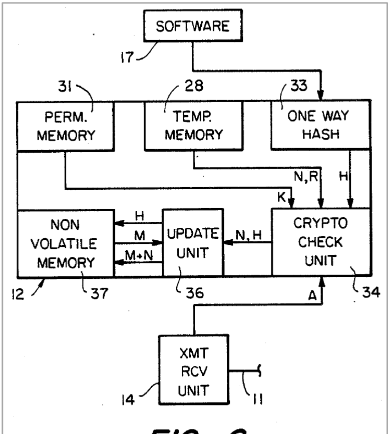


FIG_6

691. The base unit 12 has a key K stored in permanent memory 31. Hellman, 9:29-40. The permanent memory 31 can be "for example a PROM which was burned in during manufacture of the base unit." *Id*. at 9:7-10. The key K can be the same as the secret key SK used by the authorization and billing unit 13. *Id*. at 9:29-40.

692. The base unit has the values N and R stored in temporary memory 28. Hellman, 9:50-63. The temporary memory 28 can be "for example a RAM." *Id*. at 8:67-68.

693. The base unit 12 operates a cryptographic check unit 34 in order to verify the authorization A. Hellman, 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. *Id.* at 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. *Id.* at 9:50-63.

694. If the base unit determines that the authorization A is valid, then the update unit 36 accesses non-volatile memory 37. Hellman, 9:64-10:13. The base unit retrieves the value stored in the memory address represented by hash value H. *Id.* at 9:64-10:13. This retrieves the value M, which is the current number of authorized uses remaining. *Id.* at 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. *Id.* at 9:64-10:13. The non-volatile memory 37 can be "for example an EEPROM or a CMOS memory with battery backup." *Id.* at 9:64-10:13.

695. Hellman depicts the operation of the base unit 12 upon operation of software package 17 in Figure 8 (right). 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. Hellman, 10:33-43. The update unit 36 uses the hash value H "as an address to non-volatile memory 37. *Id.* at 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." *Id.* at 10:33-43.

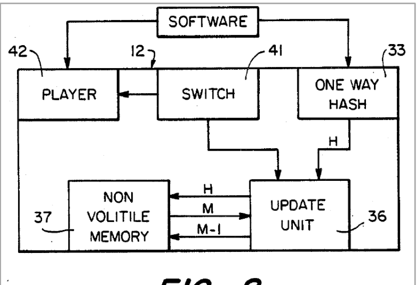


FIG_8

696. The update unit 36 then determines whether operation of the software package 17 will be permitted. Hellman, 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." *Id*. at 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." *Id*. at 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. *Id*. at 10:44-54.

697. Hellman describes a variation to the above disclosure where the software package 17 is authorized for an "unlimited number of uses." Hellman, 10:55-65. This approach can be implemented "by reserving one value of M to represent infinity." *Id*. at 10:55-65. For example, in an eight-bit value for M, the all 1's value, 255, can be reserved for "unlimited uses." *Id*. at 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." *Id*. at 10:55-65.

698. The software player 42 "will vary from application to application." Hellman, 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)." *Id*. at 10:66-11:3.

XXIX. HELLMAN IN COMBINATION WITH CHOU OR CHRISTESON INVALIDATES CLAIMS 1, 2, 6, 7, AND 11-13 OF THE '941 PATENT AS OBVIOUS

699. Claims 1, 2, 6, 7, and 11-13 of the '941 patent would have been obvious over the combination of Hellman with Chou or Christeson. I discuss below how each limitation of these claims is disclosed in the combinations of these references.

A. Claim 1 Is Rendered Obvious by Hellman in Combination with Chou or Christeson

1. Limitation 1[preamble] Is Disclosed or Rendered Obvious by Hellman, or Hellman in Combination with Chou or Christeson

700. Limitation 1[preamble] of the '941 patent recites:

> 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:

701. Hellman discloses or renders obvious this limitation. Alternatively, Hellman discloses this limitation, other than explicitly disclosing that the disclosed erasable, non-volatile memory area stores "a BIOS." Each of Chou and Christeson discloses the "erasable, non-volatile memory area of a BIOS," and that it would have been obvious to a POSITA to include the "erasable, non-volatile memory area of a BIOS" in Hellman's system if it is not inherent in Hellman's system.

702. I understand that the Court found "[a] method of restricting software operation within a license" recited in the preamble non-limiting. Thus, I understand that a prior art reference does not need to disclose this portion of the preamble to invalidate the claim. Regardless, I note that Hellman discloses restricting software operation within a license, for example, through the disclosure of the number of authorized uses value M. This value M is "the number of authorized uses of the software package ... which still remain unused prior to this new authorization." Hellman, 9:64-10:13. Hellman discloses that the value M is used to determine if operation of the software package 17 is permitted consistent with previous authorizations. *Id*. at 9:64-10:13, 10:33-49. Here, the software package 17 of Hellman may be a "computer program" or "software," and its use by the software player 42 may be "software operation." *Id*. at 10:44-49, 10:66-11:3.

703. Hellman discloses base unit 12, which satisfies the "computer" limitation. Hellman, Figures 1, 6. Hellman discloses that the base unit 12 can be a computer – "the base unit (computer, videogame base unit, record player, videorecorder or video disk player)." *Id*. at Abstract. Hellman also refers to "acquaintances with similar base units (computer)." *Id*. at 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)." *Id*. at 10:66-11:3. Thus, the base unit 12 of

Hellman satisfies the "computer" limitation.

704. Hellman discloses that the base unit 12 includes temporary memory 28, which satisfies the "volatile memory area" limitation. Hellman 8:66-67, Figure 6. Hellman discloses that the temporary memory 28 can be provided as RAM, which is a volatile memory. *Id*.

705. Hellman discloses that the base unit 12 also includes non-volatile memory 37. Hellman, Figure 6, 9:64-10:13. Hellman discloses that the non-volatile memory 37 can be provided as "an EEPROM or a CMOS memory with a battery backup." *Id*. at 9:64-10:13. A POSITA would have recognized that EEPROM is an acronym for electrically erasable programmable read-only memory. Thus, the non-volatile memory 37 satisfies the "erasable, non-volatile memory area" limitation.

706. Hellman does not explicitly disclose that the non-volatile memory 37 is a memory of a BIOS or stores a BIOS or that base unit 12 has a BIOS. However, as discussed below, a POSITA would have recognized that the base unit 12 of Hellman, implemented as a computer, inherently includes a BIOS.

707. Also, Chou discloses a computer with BIOS and an "erasable, non-volatile memory area of a BIOS." For example, Chou discloses a "BIOS EEPROM 15." Chou, Figure 1, 3:21-28. Chou discloses that the BIOS memory 15 "may be a flash EEPROM containing the various executable BIOS routines." *Id*. at 3:52-55. Chou discloses the "BIOS Memory" containing various BIOS routines. *Id*. at Figures 3, 7. Chou disclosed that there had been a transition from using other types of memory for BIOS to the use of EEPROM. *Id*. at 1:63-2:7. Based at least on these disclosures, the BIOS EEPROM 15 of Chou satisfies the "erasable, non-volatile memory area of the BIOS" limitation.

708. Christeson also discloses a computer with an "erasable, non-volatile memory

area" that stores a BIOS in the form of a FLASH memory, e.g., FLASH memory 103. Christeson, FIGs. 1 and 2; 1:40-44; 3:64-4:5; 4:19-61. As noted above, Hellman discloses that the non-volatile memory 37 can be provided as "a CMOS memory with a battery backup." Hellman, 9:64-10:13. Christeson discloses that it would be preferable to implement a partitioned FLASH memory device with separately programmable and erasable blocks of memory to back up a battery-backed CMOS memory, and that system software like BIOS is stored in FLASH memory 103. Christeson, FIGs. 1 and 2; 4:9-18. Based at least on these disclosures, the erasable block of FLASH memory 103 storing BIOS as disclosed in Christeson satisfies the "erasable, non-volatile memory area of the BIOS" limitation. Further, given the disclosed preference to back up a battery-backed CMOS memory (a disclosed embodiment of the non-volatile memory 37 of Hellman) with a FLASH memory and to store system software like BIOS in the FLASH memory, it would have been obvious for a POSITA to implement Christeson's FLASH memory in Hellman's system and store system software like BIOS there.

709. A POSITA would have found it obvious to include BIOS in the base unit 12 of Hellman. Hellman does not explicitly disclose BIOS routines, or a memory storing BIOS routines. However, given the context of Hellman, including the time period when Hellman was filed, I do not believe that the lack of explicit disclosure of BIOS in Hellman's base unit 12 means Hellman's base unit 12 is incompatible with BIOS or excludes BIOS.

710. While BIOS existed in 1983 when Hellman was filed, BIOS was not ubiquitous in all computers at that time. In contrast, around the time of the May 1998 priority date of the '941 Patent (or the earliest invention date in 1997 claimed by Ancora), a POSITA would have been aware that BIOS was ubiquitous and inherent in a computer, a disclosed embodiment of the base unit 12 in Hellman. Indeed, the applicant for the '941 Patent argued during prosecution of

the ’941 Patent that “all computer must have a BIOS ...” ...” ’941 File History, Amendment dated February 5, 2002, page 7. As a result, in the 1997 and 1998 time period, a POSITA would have found BIOS inherent in Hellman’s system, or alternatively would have found it obvious to implement Hellman’s system in a computer having a BIOS and thus a memory storing a BIOS, like the BIOS EEPROM 15 of Chou or FLASH memory 103 of Christeson.

711. When implementing BIOS in the base unit 12 of Hellman, a POSITA would have found it obvious to use the non-volatile memory 37 of Hellman to store the BIOS routines in view of Chou, or to implement the FLASH memory storing BIOS disclosed in Christeson. A POSITA would have been motivated to store the BIOS routines in the non-volatile memory 37 of Hellman, or to implement Chou’s BIOS EEPROM or Christeson’s FLASH memory storing BIOS in Hellman’s system.

712. A POSITA 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 (EEPROM) disclosed by Chou as being advantageous for storing BIOS. Hellman discloses that non-volatile memory 37 could be provided as EEPROM. Hellman, 9:64-10:13. Hellman does not disclose any other EEPROM modules. *Id*. at 8:61-9:15. Chou discloses that a transition had occurred to using EEPROM for BIOS, and that the programmability of EEPROM made it an advantageous medium for storing BIOS. Chou, 1:63-2:7. Based at least on those disclosures, a POSITA would have been motivated to store the BIOS in the non-volatile memory 37 (i.e., EEPROM) of Hellman or implement Chou’s BIOS EEPROM in Hellman’s system.

713. A POSITA would have had a strong expectation of success in storing BIOS in the non-volatile memory 37. First, Chou already describes storing BIOS in EEPROM, which was one of the formats disclosed for use for non-volatile memory 37. Hellman, 9:6410:13; Chou,

1:63-2:7, 3:21-28, 3:62-67.

714. Second, a POSITA would have been aware that EEPROM modules in the 1997 and 1998 time period had sufficient space to store both the authorization values M (which Hellman discloses as storing in the non-volatile memory 37 and being as small as 8 bits in some embodiments) and the BIOS routines from Chou. Also, Chou discloses storing add-on information and security modules in a single EEPROM with BIOS routines. Chou, 3:62-4:5, Figures 3, 7. A POSITA would have recognized that there were numerous sizes of EEPROM modules available in the 1997 and 1998 time period with sufficient space to store the BIOS routines and the authorization values from Hellman.

715. Alternatively, a POSITA would have been motivated to implement Christeson's FLASH memory in Hellman's system and store BIOS there. Hellman discloses that the non-volatile memory 37 can be provided as "a CMOS memory with a battery backup." Hellman, 9:64-10:13. Christeson discloses that it would be preferable to implement a partitioned FLASH memory device with separately programmable and erasable blocks of memory to back up a battery-backed CMOS memory, and that system software like BIOS is stored in FLASH memory 103. Christeson, FIGs. 1 and 2; 4:9-18. Based at least on these disclosures, a POSITA would have been motivated to implement Christeson's FLASH memory in Hellman's system and store system software like BIOS there.

2. __Limitation 1[a] Is Disclosed or Rendered Obvious by Hellman__

716. Limitation 1[a] of the '941 patent recites:

"selecting a program residing in the volatile memory."

717. Hellman discloses this limitation and alternatively renders it obvious based on knowledge of skill in the art.

718. Hellman discloses software package 17, which satisfies the "program" limitation.

Hellman, 5:51-56, 10:50-54, 10:66-11:3. Hellman also discloses a temporary memory 28, which satisfies the "volatile memory" limitation. Hellman, 8:61-9:15.

719. I understand that the Court construed the "selecting a program residing in the volatile memory" step can occur at any time in relation to the other method steps of claim 1. Under this construction, Hellman discloses several ways of selecting the software package 17. Hellman either discloses explicitly, discloses implicitly, or renders obvious that this selection would be of the software package 17 in the temporary memory 28.

720. First, Hellman discloses purchasing the software package 17 "at a store, over telephone line, or in some similar manner." Hellman, 5:51-56. A POSITA 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.

721. Second, Hellman discloses that the software package 17 is selected when the software package 17 is input into the one-way hash function generator 33. Hellman, 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. *Id.* at 9:16-28, 9:50-63, 10:33-49. A POSITA 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 POSITA 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 POSITA 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.

722. Third, Hellman discloses that the software package 17 is selected when the software package 17 is selected for use by the software player 42. Hellman, 10:33-11:3. Hellman discloses that software player 42 can "use software package 17." *Id*. at 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)." *Id*. at 10:66-11:3. A POSITA 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.

3. <u>Limitation 1[b] Is Disclosed or Rendered Obvious by Hellman in Combination with Chou or Christeson</u>

723. Patent Limitation 1[b] of the '941 patent recites:

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

724. Hellman in combination with Chou or Christeson as discussed above, discloses this limitation.

a. <u>"using an agent"</u>

725. I understand that the Court construed an "agent" to mean a "software program or routine." Under this construction, Hellman discloses an "agent" in the form of update unit 36. As described below, update unit 36 applies interrogatory signal representing hash value H to non-volatile memory 37 and also stores the updated authorized use value M in the nonvolatile memory 37. Hellman, 9:64-10:13. Update unit 36 retrieves the authorized use value M from the non-volatile memory 37. *Id*. at 10:33-43. Update unit 36 updates the value M in the non-volatile

memory 37. *Id*. at 10:44-49. While Hellman does not specifically disclose that update unit 36 is implemented by a software program or routine, a POSITA would have recognized that the update unit 36 would have been implemented by a "software program or routine," or it would have been obvious for a POSITA to implement the update unit 36 as a software program or routine, for example, at the operating system level. Based at least on these disclosures, a POSITA 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 as discussed below.

726. A POSITA would have additionally recognized that the authorization and billing unit 13, or the "software program or routine" implemented therein, 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 secret keys in memory 18. Hellman, 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. *Id*. at 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 SOFTWARE NAME provided in the request, and software package 21 is identical to software package 17. *Id*. at 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 POSITA would have recognized that the combination of the update unit 36 and the authorization and billing unit 13, or the software program or routine implemented therein, satisfies the "agent" limitation.

b. "set up a verification structure in the erasable, non-volatile memory of the BIOS"

727. I understand that the Court construed "set up a verification structure" to mean "establishing or certifying the existence of a pseudo-unique key and establishing at least one

license-record location." Hellman discloses a "pseudo-unique key" in the permanent memory 31 in the form of key K stored in permanent memory 31. Hellman, 9:29-40. Hellman discloses that the key K can be the same as the secret key SK. *Id*. Hellman discloses that "no two users share the same secret key." *Id*. at 9:41-45. However, Hellman does not specifically disclose "establishing or certifying the existence of a pseudo-unique key" in the non-volatile memory 37. As detailed above, it would have been obvious to a POSITA to implement the BIOS EEPROM 15 of Chou in Hellman's system. Chou discloses that the BIOS EEPROM 15 stores a public key. Chou, Figure 3, 4:6-27. As discussed above in connection with the Arbaugh Patent, I understand that Ancora asserts that the limitation of a "pseudo-unique key" can be met by a public key used to decrypt data encrypted by another device. See Ancora Final Infringement Contentions, Ex. B, at 105 ("For example, LGE includes a public key in the hardware and/or non-volatile cache memory of each Accused Android Product . . ."). Under this interpretation of the "pseudo-unique key" by Ancora, Hellman in combination with Chou's BIOS EEPROM 15 with a public key discloses a pseudo-unique key in the erasable, non-volatile memory of the BIOS.

728. Hellman discloses setting a "verification structure" in the form of storing at least one value M at memory addresses or a location defined by at least one hash value H in non-volatile memory 37. 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. Hellman, 6:16-61. Hellman discloses that hash value H has the characteristic that "it is easily com[]puted 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." *Id*. at 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." *Id*. at 9:64-10:13, 10:33-43. Regarding the "erasable, non-volatile memory of the BIOS" limitation, see the above discussion with respect to Limitation 1[preamble].

729. Based at least on these disclosures, and under the Court's construction of this limitation and Ancora's interpretation of the "pseudo-unique key" as discussed above, the above combination of Hellman with Chou discloses this limaition.

c. <u>"the verification structure accommodating data that includes at least one license record"</u>

730. I understand that the Court construed "license record" to mean "data associated with a licensed program with information for verifying that licensed program." 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." Hellman, 9:64-10:13. Thus, it is associated with the software package, the licensed program. Hellman further discloses that the value M is used to determine if operation of the software is permitted consistent with previous authorizations. *Id*. at 9:64-10:13, 10:33-49. The value of M is increased based on payment for uses made by the user. *Id*. at 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, the value M is associated with the software package and has information used to verify that the software package is licensed for use in the base unit 12. Similarly, the '941 Patent discloses that a license record can include a "number of licensed users" or a "number of licensed copies." '941 Patent, 1:53-58, 5:25-33. Thus, a POSITA would have recognized that the value M is a "license record," and that the verification structure discussed above in the non-volatile memory 37

accommodates the value M.

4. ***Limitation 1[c] Is Disclosed or Rendered Obvious by Hellman, or Hellman in Combination with Chou or Christeson***

731. Limitation 1[c] of the '941 patent recites:

> verifying the program using at least the verification structure from the erasable non-volatile memory of the BIOS.

732. Hellman discloses this limitation. As discussed above, Hellman discloses the "program" (software package 17), the "verification structure" (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" (the non-volatile memory 37, inherently or in combination with the teachings of Chou or Christeson).

733. 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. Hellman, 10:33-54. The base unit then uses hash value H as a memory address to non-volatile memory 37 in order to retrieve the number of authorized uses M for that software package 17. *Id*. at 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. *Id*. at 10:33-54. Based at least on these disclosures, a POSITA would have recognized that Hellman discloses this limitation.

5. ***Limitation 1[d] Is Disclosed or Rendered Obvious by Hellman***

734. Limitation 1[d] of the '941 patent recites:

> acting on the program according to the verification.

735. Hellman discloses this limitation. As discussed above, Hellman discloses the "program" (software package 17). Hellman also discloses "the verification" (the "verifying" step described for Limitation 1[c]).

736. Hellman discloses acting on the program according to the verification in the form of preventing use of the software package 17 by the software player 42 if the number of authorized uses is zero. Hellman, 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. *Id*. at 10:44-49. However, if the value of M is zero, then the update unit 36 does not activate the switch and prevents the software player 42 from using software package 17. *Id*. at 10:50-54 ("The user is thus prevented from using software for which he does not have current authorized use."). This satisfies the limitation.

B. Claim 2 Is Rendered Obvious by Hellman in Combination with Chou or Christeson

737. Limitation 2 of the '941 patent recites:

> A method according to claim 1, further comprising the steps of: establishing a license authentication bureau.

738. As discussed above, claim 1, from which claim 2 depends, is rendered obvious by Hellman in combination with Chou or Christeson. In addition, Hellman discloses the step of "establishing a license authentication bureau" in dependent claim 2.

739. 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. Hellman, 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." *Id*. at 4:46-63. As discussed above, a POSITA would have recognized that the number of authorized uses in Hellman represents a license to use software package 17 in the base unit 12. *Id*. at 6:3-15, 9:64-10:13.

740. 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 POSITA would have recognized that Hellman discloses "establishing a license authentication bureau."

C. Claim 6 Is Rendered Obvious by Hellman in Combination with Chou or Christeson

741. Claim 6 of the '941 Patent recites:

> 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.

742. As discussed above, claim 1, from which claim 6 depends, is rendered obvious by Hellman in combination with Chou or Christeson. In addition, Hellman discloses the additionally recited limitations of dependent claim 6.

743. As discussed above, Hellman discloses the "computer" (base unit 12), the "volatile memory" (temporary memory 28), a "licensed-software-program" (software package 17), and the "license-record" (authorized uses value M).

744. As discussed for Limitation 1[a], Hellman discloses loading the software package 17 in the temporary memory 28. Based at least on these disclosures, a POSITA would have recognized that Hellman discloses "establishing a licensed-software-program in the volatile memory of the computer."

Further, Hellman discloses that the entirety of the software package 17 is used to generate the hash value H. Hellman, 6:31-61, 9:16-28. The hash value H is part of the authorization A. *Id*. at 6:62-7:2. As discussed above with respect to Limitation 1[b], the authorization A, which includes the hash value H, leads to the updating of the authorized use value M in the non-volatile memory 37 and therefore is used to form the value M. Based at least on these disclosures, a

POSITA would have recognized that Hellman discloses that the "licensed-software-program includes contents used to form the license-record."

D. Claim 7 Is Rendered Obvious by Hellman in Combination with Chou or Christeson

1. *Limitation 7[a] Is Disclosed or Rendered Obvious by Hellman*

745. Limitation 7[a] of the '941 patent recites:

> A method according to claim 6 wherein using an agent to set up the verification structure includes the steps of: establishing or certifying the existence of a pseudo-unique key in a first non-volatile memory area of the computer.

746. As discussed above, claim 6, from which claim 7 depends, is rendered obvious by Hellman in combination with Chou or Christeson. With respect to the additionally recited portion of Limitation 7[a], see the above discussion with respect to Limitation 1[b].

2. *Limitation 7[b] Is Rendered Obvious by Hellman, or Hellman in Combination with Chou or Christeson*

747. Limitation 7[b] of the '941 patent recites:

> establishing at least one license-record location in the first nonvolatile memory area or in the erasable, nonvolatile memory area of the BIOS.

748. As discussed above, e.g., with respect to Limitation 1[preamble], a combination of Hellman and Chou discloses the "non-volatile memory area of the BIOS" in the form of non-volatile memory 37, as modified by Chou's BIOS EEPROM. Alternatively, as also discussed above, a combination of Hellman and Christeson discloses implementing FLASH memory storing BIOS in Hellman's system.

749. Hellman further 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) for the corresponding software package 17 will be stored. Hellman, 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. *Id*. at 9:64-10:13. And hash value H is generated based on the content of the software package itself. *Id*. at 31-61, 9:16-28. As such, a POSITA would have recognized that Hellman discloses establishing a license record location in the non-volatile memory 37.

750. Based at least on these disclosures, a POSITA would have recognized that Hellman, or Hellman in combination with Chou or Christeson, 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."

E. Claim 11 Is Rendered Obvious by Hellman in Combination with Chou or Christeson

751. Claim 11 of the '941 patent recites:

A method according to claim 1 wherein the volatile memory is a RAM.

752. As discussed above, claim 1, from which claim 11 depends, is rendered obvious by Hellman in combination with Chou or Christeson. In addition, Hellman discloses the additionally recited limitation of claim 11.

753. As discussed above for Limitation 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." Hellman, 8:67-68.

F. Claim 12 Is Rendered Obvious by Hellman in Combination with Chou or Christeson

754. Claim 12 of the '941 patent recites:

A method according to claim 1, wherein a pseudo-unique key is stored in the non-volatile memory of the BIOS.

755. As discussed above, claim 1, from which claim 12 depends, is rendered obvious

by Hellman in combination with Chou or Christeson. In addition, Hellman in combination with Chou discloses the additionally recited limitation of claim 12.

756. As discussed above in connection with Limitation 1[b], under Ancora's interpretation of a "pseudo-unique key" in its Final Infringemetn Contentions, Hellman in combianation with Chou's BIOS EEPROM discloses this additional limitation of claim 12.

G. Claim 13 Is Rendered Obvious by Hellman in Combination with Chou or Christeson

757. As discussed above, claim 1, from which claim 13 depends, is rendered obvious by Hellman in combination with Chou or Christeson. With respect to the additionally recited limitation in claim 13, see the above discussion with respect to Limitation 1[b].

XXX. SECONDARY CONSIDERATIONS

758. It is my opinion that the asserted claims of the '941 patent would have been obvious to persons of ordinary skill in the art at the time of the alleged invention notwithstanding Ancora's claimed secondary considerations. As explained above, I am informed that secondary considerations of nonobviousness can include, among other things, commercial success; skepticism of others; long-felt but unmet need for the alleged invention; praise; unexpected results; and failure of others. I am informed that it is the patent holder's burden to come forward with evidence of secondary considerations if it believes that any apply to an obviousness analysis. I am informed that if the patentee provides evidence of secondary considerations, and they are tied to the claimed inventions of the asserted claims, then they are considered in the analysis of whether the claims are obvious. I am informed that the law requires the scope of the secondary considerations to be commensurate with the scope of the claims and to have a "nexus" to the claimed invention, meaning they must be tied to the claimed invention rather than to something already in the prior art. I am informed that the patentee must establish a link between

the evidence of the asserted secondary consideration and the novel, claimed features of the alleged invention.

759. I was asked to consider Ancora's contentions about secondary considerations of nonobviousness. I have reviewed Ancora's response to Defendants' Interrogatory No. 5, which asked Ancora to identify and describe in detail each secondary consideration or other objective evidence of nonobviousness that Ancora contends apply to each asserted claim of the '941 patent. Ancora listed (1) determinations of validity at the patent office, during the covered business method ("CBM") patent review, and at the Federal Circuit, (2) a long-felt but unmet need, (3) commercial success and commercial acquiescence, (4) Ancora's licenses to the patent, the lack of near simultaneous development of the invention, (6) prior art references teaching away, and (7) Microsoft's adoption of the patented invention, which provides evidence of copying. Having reviewed Ancora's response, I do not see any support for Ancora's assertion that the listed secondary considerations apply to the claims of the '941 patent.

760. I do not believe there is support for Ancora's argument that the '941 patent is nonobvious because the patent has survived United States Patent and Trademark Office review twice as well as a CBM patent review. I reviewed all documents cited in Ancora's response to Interrogatory No. 5, including the PTO's decision denying CBM patent review. *HTC Corp. v. Ancora Tech., Inc.*, Decision Denying Institution of Covered Business Method Patent Review, 2017 WL 6032605 (Patent Tr. & App. Bd. Dec. 1, 2017). First, I am informed that courts do not consider Ancora's argument about PTO review to be a secondary consideration. Second, a reexamination proceeding can be only as effective as the prior art references submitted to the PTO, and I am informed that the PTO has never had the opportunity to review the '941 patent in light of Arbaugh and Jablon. ANCORA_00000450-52. And third, the PTO's decision to deny

CBM patent review did not consider obviousness. Additionally, I am informed that the Federal Circuit has never analyzed the issue of obviousness. Ancora therefore cannot take the Federal Circuit's general background statements on the patent to support nonobviousness arguments never presented to that court. *Ancora Techs., Inc. v. HTC Am., Inc.*, 908 F.3d 1343 (Fed. Cir. 2018).

761. I have not seen any facts to suggest that there was a long-felt but unmet need in the art for the specific solution claimed by the '941 patent, and I do not believe there is support in the prior art Ancora cited. DEFS_ANCORA00000129 – DEFS_ANCORA00000144; DEFS_ANCORA00000222 – DEFS_ANCORA00000237; DEFS_ANCORA00000256 – DEFS_ANCORA00000280; DEFS_ANCORA00000607 – DEFS_ANCORA00000629; DEFS_ANCORA00000695 – DEFS_ANCORA00000798 .

762. Moreover, Ancora's response to Interrogatory No. 5 attempts to generalize the problem as "providing a secure and effective method of restricting unauthorized software without relying on a separate hardware device or hardware component to verify that a program is authorized to run on a computer." But the solution in the '941 patent is much more specific than this general description. A number of methods for verifying software predate the patent that do not use a separate hardware device.

763. Further, I have not seen any evidence showing commercial success or commercial acquiescence. I am informed that Ancora bears the burden of production to come forward with evidence of secondary considerations, and it has not produced any evidence that, as Ancora argues in its response to Interrogatory No. 5, "the largest participants in the market for smartphones, tablets, and smart TVs have adopted and incorporated the invention claimed in the '941 patent." It appears that Ancora equates filing suit against the largest participants in the

market—alleging patent infringement—as evidence that these market participants ”have adopted and incorporated the invention.” But I have seen no evidence to support Ancora’s argument that these market participants use the ’941 patent method. I have also seen no evidence that there exists a nexus between any alleged commercial success and the ’941 patent.

764. I have also not seen any evidence that Ancora’s licenses show “a recognition and acceptance of commercial acquiescence of the ’941 patented technology,” as Ancora argues. I am informed that the only two licenses that Ancora has produced are litigation settlement licenses, and I do not believe settlement agreements show acceptance of commercial acquiescence. Ancora failed to provide evidence showing that either Microsoft or Apple practiced the claims of the ’941 patent.

765. Ancora also lists the lack of simultaneous or near simultaneous development of the invention, but as I explain above, my review of the prior art—including the Arbaugh and Jablon patents—demonstrate that others conceived of and reduced to practice identical or similar claims, as Ancora has interpreted the claims of the ’941 patent, even before the invention of the ’941 patent. And even co-Defendant Samsung’s patenting a similar but distinguishable invention after the priority date for the ’941 patent, ANCORA_00003406, does not change the fact that other inventors had already invented the same method, rendering the ’941 patent obvious at minimum. Moreover, contrary to Ancora’s conclusory arguments in its response to Interrogatory No. 5, at least Arbaugh and Jablon do not teach away from “allowing programs running at the operating-system level to interact with a program verification structure stored in the BIOS to verify a program.” Indeed, that is the very teaching of the Arbaugh and Jablon patents, both of which antedate the ’941 patent.

766. Ancora also lists Microsoft’s adoption of the patented invention “after receiving a

disclosure of the ’941 patent” as evidence of copying. Ancora does not provide any evidence, however, that Microsoft adopted the patented invention. And the document Ancora provides as evidence that Microsoft received a disclosure of the ’941 patent in 2003 establishes only that one Microsoft employee received from Beeble (which I am informed is the predecessor to Ancora)

767. “unsolicited [marketing] materials” that the employee was “not at liberty to review” that briefly reference and explain the ’941 patent. ANCORA_00003911–18. These “unsolicited materials” were sent on to Microsoft’s internal patent counsel, and there is no evidence there were any further communications. I do not conclude from this that Microsoft copied Ancora’s method or credit this evidence as demonstrating nonobviousness. On the contrary, I conclude from the lack of evidence of copying that market participants did not copy Ancora’s technology but merely adopted a method of verifying software already known in the art or obvious to persons of ordinary skill in the art.

768. Considering what was known in the art prior to the ’941 patent and Ancora’s alleged secondary considerations, it is my opinion that the asserted claims of the ’941 patent would have been obvious to a person of ordinary skill in the art at the time of the alleged invention.

XXXI. INCREMENTAL VALUE OF THE ALLEGED INVENTION OF THE ’941 PATENT

769. I have been asked to present my opinion on the incremental value of the alleged invention of the ’941 Patent over the state of the prior art, if any. As I have opined earlier, the prior art that I rely on for my opinion renders the asserted claims of the ’941 Patent invalid. I understand that, nonetheless, Ancora asserts that the alleged innovation of the ’941 Patent renders it patentably distinct from the prior art. My opinion here addresses the incremental value of that alleged innovation, notwithstanding my opinion that the ’941 Patent is invalid over the

prior art I rely on herein.

770. The specification of the ’941 Patent conceded that there were already known approaches to securing software using licenses, such as by simply storing a license on a hard disk (which the ’941 Patent inventor erroneously called “volatile memory”):

> Software based products have been developed to validate authorized software usage by writing a license signature onto the computer's volatile memory (*e.g.* hard disk). These products may be appropriate for restricting honest software users, but they are very vulnerable to attack at the hands of skilled system's programmers (*e.g.* “hackers”). These license signatures are also subject to the physical instabilities of their volatile memory media.

Id., at 1:19-26. I agree that this approach was well-known.

771. A second approach the ’941 Patent concedes was well-known was the use of hardware products to authorize software, by appending a hardware device to a computer in order to validate software:

> Hardware based products have also been developed to validate authorized software usage by accessing a dongle that is coupled *e.g.* to the parallel port of the P.C. These units are expensive, inconvenient, and not particularly suitable for software that may be sold by downloading (*e.g.* over the internet).

Id., at 1:27-32. I am familiar with these techniques, and a POSITA would understand that a dongle provides, in effect, a unique token that can be used to verify operation. The software can attempt to query or read a value from the dongle, and if the received signal is different from what is expected, the software will not execute.

772. The alleged improvement of the ’941 Patent is that a unique value is provided, for example by inserting a unique key into the non-modifiable BIOS of the computer at the time of manufacture. *Id.*, at 1:44-52. This key is then used to encrypt license information to yield a license, which is then stored in modifiable memory. *Id.*, at 1:65-2:9. The ’941 Patent also states that storing a license record in the BIOS of the computer was advantageous of the higher

programming expertize needed to intercept and modify commands interacting with the BIOS. *Id.*, at 3:4-17.

773. As discussed above, during prosecution the pending claims were rejected as obvious based on a combination of the Misra patent, the Goldman patent and Ewertz patent. '941 File History, at 197. According to the examiner, though Misra did not teach that the keys were pseudo-unique or storing the records within the BIOS, Goldman taught a pseudo-unique key and Ewertz taught using BIOS memory to store data. *Id.* at 199. In other words, the examiner notes that all the elements of the pending claim 1 were known in the prior art.

774. In response, the applicant admitted that several of the elements were disclosed by the Misra patent, it argued that a "license ID", an element that uniquely identifies licensed software, is never stored in the computer BIOS. *Id.*, at 207. The applicant then used this distinction to argue that Ewertz could not be combined to teach storing the license ID of Misra in the BIOS, because the only data Ewertz teaches being stored in the BIOS corresponds to the client system ID of Misra. *Id.* at 208. The examiner then allowed the patent on March 28, 2002, noting, among other things, the prior art did not teach using BIOS in the manner claimed by the applicant. *Id.*, at 217 (Notice of Allowance, March 28, 2002). Thus, the alleged invention of the '941 patent is very narrow. Its alleged innovative aspect is simply the use of BIOS to store the license records.

775. I understand that the inventor of the '941 patent provided a tutorial during Ancora's litigation. The inventor summarized the alleged invention of the '941 patent as follows:

> In – I had a conversation, discussions with a friend of mine who is a hardware engineer and who is a co-inventor on this patent, Julian, and together we've – we identified that there's one standard component on the motherboard of the computer that we could potentially reuse and repurpose, and build a third class of software licensing solution. That component is called BIOS, and I explain in a couple of minutes what it is.

> Our idea essentially is – was to build a software licensing technology that combines this already available hardware with software that we will develop, and create an elegant solution that has the advantages with none of the disadvantages. We spent the better of 1997 developing a prototype proving to ourselves that this technology can actually work.

Transcript of Ancora-Apple Markman Tutorial, pp. 6-7.. The inventor's tutorial confirms that the innovative aspect of the '941 patent is limited to leveraging pre-existing BIOS.

776. During his inventor deposition in the Apple litigation, Mr. Mullor framed the innovation of the '941 patent as teaching verification based on storage of a license in the BIOS of the computer:

> Q. Were you the first person to think of verifying a program based on a license that was stored somewhere?
>
> A. As I said, I was the first one with Julian. We were the first ones to think of verifying a program with license that we stored in the BIOS of the computer.
>
> Did other people do other things?
>
> Yes.

Miki Mullor Dep. Tr. (Oct. 15, 2015) at 54:7-15.

777. Mr. Mullor then, in his response to the next question, acknowledged that the prior art taught verifying a program based on a license:

> Q. Before you and Julian, Mr. Vlaiko, filed for your patent, there were other people in the world that were verifying a program based on a license that was stored not in BIOS memory?
>
> A. Yes.

Miki Mullor Dep. Tr. (Oct. 15, 2015) at 54:18-22.

778. These consecutive statements – first framing the innovation of the '941 patent as verifying a program based on a license stored in BIOS, then acknowledging that all these steps except for storage in the BIOS were known in the prior art – demonstrates that the innovation of the '941 cannot extend beyond storage in the BIOS.

779. Mr. Mullor also acknowledged during his deposition that the prior art taught a BIOS with an erasable nonvolatile memory area, see *id.* at 52:2-9; selecting a program that resides in volatile memory, *id.* at 52:13-18; the use of an agent, *id.* at 52:20-53:1; a structure accommodating data, *id.* at 53:7-9; and acting on a program according to a verification, *id.* at 56:22-57:9. These admissions further demonstrate that the alleged innovation of the '941 patent is limited to leveraging pre-existing BIOS.

780. This alleged improvement is not significant. As the '941 Patent specification concedes, the license information itself, stored in a modifiable part of the BIOS, can still be read and written-to by a person trying to violate the license. *Id.*, at 2:37-58. Therefore the use of the BIOS to store the license record does not actually improve security, because it can still be read from and written to. The entire benefit of the alleged improvement therefore depends on having an immutable key that uniquely identifies a computer on which the software is to be run, because this is the only remaining part of the invention that serves the asserted purpose:

781. It is important to note that the hacker is unable to modify the key in the ROM of the second computer to K1, since, as recalled, the contents of the ROM is established during manufacture and is practically invariable. *Id.*, at 2:43-47. The claims do not, however, require that the claimed "key" or any part of the verification structure is stored in a read-only memory. The claims, in fact, allow that the key used to generate and verify licenses can be changed, because they are stored in a non-volatile memory which, absent any further qualifier, may be erasable.

782. Because the alleged invention as-claimed does not meet any of the stated benefits of the alleged invention over the prior art, within the four walls of the '941 Patent itself the inventor has shown there is no meaningful value over that prior art. Even if the alleged invention

as-claimed had some benefit, however, the value of that benefit over the prior art would be minimal because there were an ample number of alternative ways to provide a verification of the ability to execute software without using the alleged invention of the ’941 Patent, including the use of the techniques described in the prior art admitted in the specification of the ’941 Patent, or in references like the Jablon Patent.

XXXII. CONCLUSION

783. I have reviewed certain documents and information identified in Appendix B, which I have considered and which form part of the basis for my analysis and opinions in this report. To the extent not identified in Appendix B, I have also considered and am relying on each of the documents cited in this report and its other exhibits as part of the basis for my analysis and opinions. I am also relying on my expert qualifications and experience for my analysis and opinions.

784. The opinions and analysis in this report are based upon my consideration of the materials I have reviewed to date. Should additional information become available through discovery or at trial, or should the Court adopt a claim construction I have not considered, the information and opinions presented herein may be supplemented or amended.

785. If I am called upon to testify about this report by deposition or at a hearing before the Court, I may cite other documents or information similar to that specifically identified in this report. I may also use graphics, animations, pictures, demonstrations, and/or other audio/visual aids to explain my analysis and opinions.

786. I have personal knowledge of the facts set forth in this Report, I am competent to testify to all matters stated herein and, if called upon, would testify to the same.

I declare under the penalty of perjury that the foregoing is true and correct.

[illegible]

Dated: January 22, 2021 __________________________________

Suzanne Barber

APPENDIX A

K. SUZANNE BARBER

EDUCATION:

Degree	*School*	*Date*
B.S. Engineering Science	Trinity University	1985
M.S. Electrical Engineering	The University of Texas at Arlington	1988
Ph.D. Electrical Engineering	The University of Texas at Arlington	1992

ACADEMIC APPOINTMENTS:

The University of Texas at Austin

AT&T Endowed Professor, Electrical and Computer Engineering Department **September 2002 - present**

Associate Professor, Electrical and Computer Engineering Department **September 1997 – August 2002**

Assistant Professor, Electrical and Computer Engineering Department **September 1992 – August 1997**

Teaching portfolio includes: Information Security and Privacy, Identity Management and Security, Information Repositories, Introduction to Programming, Requirements Management, Software Architectures and Software Design.

Founding Director, Center for Identity **September 2010 – present**

The Center for Identity serves as a research center of excellence empowering individuals and organizations to make well-informed and intentional decisions with regard to the personal data they collect, use, share, and protect with the aim of increasing trust, convenience security and privacy. The Center is a public-private partnership bringing together the depth and breadth of knowledge and talent at The University of Texas with partners from corporations, government, and academia. Responsible for strategic vision, fund raising, fiscal oversight, recruiting, and building partnerships with faculty, corporate, and government leaders.

Founding Director, Masters Degree in Information Security and Privacy (MSISP) **September 2015 – present**

(formerly the MS Degree program in Identity Management and Security (MSIMS)

This M.S. degree program is offered by the School of Information at The University of Texas, ranked Top 5 in the Nation, in conjunction with the Center for Identity. The MSISP program aims to educate professionals who are responsible for information security and privacy at all levels of responsibility by offering a holistic, interdisciplinary education ensuring that professionals from multiple market sectors, roles, and levels of responsibility acquire the knowledge and skills necessary to be effective stewards of information and leaders of technological, policy, legal and societal initiatives. Responsible for strategic vision, academic curriculum, scheduling, fiscal planning, budget oversight, and student academic counseling.

Director, Software Engineering **April 2004 – September 2010**

Directed the Software Engineering academic and research programs to include faculty recruiting and mentorship, served Director of Software Engineering Research Center, and served Director of M.S. Degree Program for working professionals.

Director, The Laboratory for Intelligent Processes and Systems (UT:LIPS) **September 1992 – present**

Originally established as a personal research group and now a laboratory within the Center for Identity, UT:LIPS focuses on the following research areas: 1) online trust and cyber security, 2) formal systems and software systems analysis for design methods and tools; and, 2) distributed, intelligent decision-support systems. Federal agencies, state agencies, and industrial concerns have invested in the development and application of UT:LIPS research for identity management, privacy, cyber-security, cyber-trust, social networks, e-commerce, first responder emergency response, bio-surveillance for epidemics, UAV reconnaissance, military command and control, radar interference management, maritime domain awareness, and intelligence operations for the global war on terror. Responsible for funding acquisition, strategic research planning, research investigations, graduate student oversight, and publications.

Founding Director, Executive Program in Software Engineering **December 2002 – September 2015**
September 1996 – November 1999

This program offers an advanced software engineering education with scheduling that allows working professionals to obtain a Master's Degree from the Department of Electrical and Computer Engineering. Professionals from over 200 companies have received M.S. degrees under this program since 1997. Responsible for the academic curriculum, scheduling, fiscal planning, budget oversight, and student academic counseling.

Chair of Board, Systems and Software Engineering Institute **September 1996 – September 1999**

The Institute provided short courses and a certificate program for software engineering techniques and practices in preparation for quality metrics and management. Courses offered at industrial sites or transmitted via distance learning facilities. Responsible for strategic planning, program management, and fiscal planning.

Automation and Robotics Research Institute (ARRI) **June 1986 – August 1992**
The University of Texas at Arlington
Faculty Associate

Proposed, supervised and conducted intelligent decision-support systems research funded by federal and industrial agencies. Represented ARRI in industrial and federal consortia and meetings. Contributed to research and standards for federally-sponsored Next Generation Control Project to improve robotic and machine tool manufacturer's competitiveness.

The Robotics Institute, Carnegie-Mellon University **May 1985 – June 1986**
Research Associate

Since its founding in 1979, the Robotics Institute has been leading the world in integrating robotic technologies into everyday life. Responsible for Rapidly Programmable Robotic Assembly research project that developed an interactive robotic programming language teaching robots assembly tasks through interactive instruction (i.e., human dialog) rather than manual programming.

BOARD OF DIRECTORS AND ADVISORS:

- Department of Homeland Security (DHS), Senior Advisor, February 2020– present.
- Department of Homeland Security (DHS) Data Privacy and Integrity Advisory Committee appointed by and in service to the Secretary of the Department of Homeland Security, June 2012 – December 2019.
- Metroplex Technology Business Council (MTBC), Board of Directors, Fall 2008 – 2012.
- Center for Applied Identity Management Research (CAIMR), Board of Directors, Spring 2009 – Fall 2010.
- Army Science Board.
- Defense Science Study Group, DARPA and Institute of Defense Analysis (IDA) —Consultant.
- ANCA TECH, Board of Advisors. Spring 99 – Fall 2004.
- ALVE L.C. Communications Software, Board of Advisors, Spring 99 – Fall 2004.

OTHER PROFESSIONAL EXPERIENCE:

- **Expert Witness –** Served as an expert witness for successful software-related cases in telecommunications, manufacturing, travel, and retail industries in support of law firms in Georgia, Texas, California, New York, and Washington, DC.
- **Institute for Defense Analysis (IDA), Alexandria, VA. –** Consultant for technology reviews and advice.
- **University XXI and Central Technical Support Facility (CTSF), Fort Hood, TX. –** Consultant for technology reviews and advice.
- **Asyst Technologies, Austin, TX. –** Consultant to direct requirements management efforts and support system architecture development.
- **U.S. Army Simulation Training and Instrumentation Command (STRICOM), Orlando, FL. –** Consultant to teach course regarding Systems Engineering methods and implementation strategies for web-accessible tools supporting collaborative system analysis, design, development and deployment.
- **US Army CECOM, Fort Monmouth, NJ. –** Consultant to perform system engineering analysis and design for command and control applications, components and interfaces as part of an effort to verify and validate a comprehensive architecture. Performed consultant services to assess available enterprise tools and methodologies that will improve the system engineering approach used in the development of embedded systems.
- **Applied Research Laboratory, Austin, TX. –** Consultant to assist in the specification of next generation systems architectures for Army training centers to include hardware, software and communications systems specifications to support Military Command and Control.
- **Institute of Advanced Technology, Austin, TX. –** Consultant to teach courses to Senior Army Officers regarding systems engineering methods and implementation strategies.
- **Tracor, (now BAE Systems), Austin, TX. –** Consultant to direct initial system specification and modeling effort defining Defense Logistics Systems.

- **Automation Engineering, Dallas, TX. –** Consultant providing instructional courses regarding user-centered methods for acquiring, specifying and maintaining requirements driving system design, development and deployment.
- **Texas Instruments, Dallas, TX. –** Consultant to review, evaluate and comment on products (knowledge acquisition reports, domain models and reference architecture) related to the development and deployment of military and civilian trauma care information systems.
- **National Center for Manufacturing Sciences, Ann Arbor, MI. –** Consultant to review research, government initiatives, and commercial software systems addressing planning in manufacturing environments.
- **Naval Surface Warfare Center, Dahlgren, VA. –** Consultant to teach courses regarding systems engineering methods and implementation strategies.

Refereed Archival Journal Publications

1. Zaeem, R. N., and K. S. Barber, “The Effect of the GDPR on Privacy Policies: Recent Progress and Future Promise,” ACM Transactions on Management Information Systems (TMIS) Special Issue on Analytics for Cybersecurity and Privacy. (to appear December 2020)
2. Barber, K. S., “The Future Needs Identity Professionals,” Keesing Journal of Documents & Identity p. 19, February 2020.
3. Huang, T., R.N. Zaeem, and K.S. Barber, “Identifying Real-World Credible Experts in the Financial Domain to Avoid Fake News,” ACM Digital Threats: Research and Practice (DTRAP) Special Issue on Fake News Research. (under review)
4. Barber, K. S., “Privacy in the Digital Age,” Keesing Journal of Documents & Identity p. 48, October 2019.
5. Huang, T., R.N. Zaeem, and K.S. Barber, “It Is an Equal Failing to Trust Everybody, and to Trust Nobody: Stock Price Prediction Using Trust Filters and Enhanced User Sentiment on Twitter,” ACM Transactions on Internet Technology (TOIT), 19(4), 48, June 2019.
6. Barber, S., R. Anderson, R.N. Zaeem, and J. Zaiss, “Identity Threat Assessment and Prediction (ITAP): “Analysing identity crime to prevent identity crime,” Keesing Journal of Documents & Identity, Issue 58, pp.14-19, 2019.
7. Zaiss, J., R.N. Zaeem, and K.S. Barber, “Identity Threat Assessment and Prediction,” Journal of Consumer Affairs: Trends and Applications, 53(1), pp. 58-70, 2019.
8. DeAngelis, D., R. N. Zaeem, and K. S. Barber, “Finding and Motivating Trusted Expert Participation in Online Communities,” invited submission to Applications of Agent Models and Architectures, IGI Global, 2018.
9. Nokhbeh Zaeem, R., German, R. L., and Barber, K. Suzanne, “PrivacyCheck: Automatic Summarization of Privacy Policies Using Data Mining,” ACM Transactions on Internet Technology (TOIT), 18 (4), Article 53. May 2018.
10. Nokhbeh Zaeem, R., M. Manoharan, M., Y. Yang, and K. Suzanne Barber, “Modeling and Analysis of Identity Threat Behaviors through Text Mining of Identity Theft Stories,” Journal of Computers and Security Vol. 65, pp. 50-63, 2017.
11. Budalakoti, S., R. Nokhbeh Zaeem, and K.S. Barber, “Tournament Models for Authority Identification in Online Communities,” International Journal of Computer and Information Technology (IJCIT) (ISSN: 2279 – 0764) 6 (2), pp. 75--83, 2017.
12. Nokhbeh Zaeem, R. and K. Suzanne Barber, “A Study of Web Privacy Policies Across Industries,” Journal of Information Privacy and Security 13(4), pp. 169--185, Nov. 2017.
13. DeAngelis, D. and K.S. Barber, “Systematic Reciprocal Rewards: Motivating Expert Participation in Online Communities with a Novel Class of Incentives,” *International Journal of Agent Technologies and Systems (IJATS)*, Vol. 6(2), pp. 30-50, 2014.
14. Golden, R. and K.S. Barber, “Supporting Identity Risk Identification and Analysis Through Text Mining of News Stories,” *International Journal of Computer and Information Technology (IJCIT)*, Vol. 3(5), pp. 850-859, 2014.
15. DeAngelis, D. and K.S. Barber, “Incentives in Online Communities,” *International Journal of Computer and Information Technology (IJCIT)*, vol. 3(6), pp. 1229-1240, 2014
16. Jones, C.L.D., and K. S. Barber, “Robust Coalition Formation in a Dynamic, Contractless Environment,” submitted March 2012 and under review by Journal of Autonomous Agents and Multi-Agent Systems.
17. DeAngelis, D. and K.S. Barber, “Systematic Reciprocal Rewards: Motivating Expert Participation in Online Communities with a Novel Class of Incentives,” International Journal of Agent Technologies and Systems (IJATS), Vol. 6(2), pp. 30-50, 2014.
18. Golden, R. and K.S. Barber, “Supporting Identity Risk Identification and Analysis Through Text Mining of News Stories,” International Journal of Computer and Information Technology (IJCIT), Vol. 3(5), pp. 850-859, 2014.

19. DeAngelis, D. and K.S. Barber, "Incentives in Online Communities," *International Journal of Computer and Information Technology (IJCIT)*, vol. 3(6), pp. 1229-1240, 2014.
20. Han, D. and K. S. Barber. "Simulating UAV Surveillance for Analyzing Impact of Commitments in Multi-Agent Systems," International Journal of Agent Technologies and Systems, Vol. 4, Issue 1, pp. 1-16, 2012.
21. Kadaba, R., S. Budalakoti, D. DeAngelis, and K. S. Barber. " Modeling Virtual Footprints," International Journal of Agent Technologies and Systems, IGI Publishers, Volume 3, Issue 2, pp. 1-17, 2011.
22. Kadaba, R., S Budalakoti, D. DeAngelis, and K. S. Barber, "Modeling Virtual Footprints: Impact on Security and Identity," ACM Transactions on Intelligent Systems and Technology, Special Issue on Trust in Multiagent Systems; 2011.
23. DeAngelis, D. and K. S. Barber. "Security Applications of Trust in Multi-Agent Systems," Journal of Computer Security, IOS Press. Vol. 19, Issue 1, pp. 57-99, 2011.
24. Jones, C.L.D., and K. S. Barber. "Combining Job and Team Selection Heuristics," LNCS: Coordination, Organizations, Institutions and Norms in Agent Systems IV: COIN 2008 International Workshops. p. 33-47, 2009.
25. Han, D., J. Park, K. Fullam, and K. S. Barber. "Application of Action Selection, Information Gathering, and Information Evaluation Technologies to UAV Target Tracking," Defense Applications of Multi-Agent Systems: International Workshop, DAMAS 2005, Utrecht, The Netherlands, July 25, 2005, Revised and Invited Papers, S. Thompson and R. Ghanea-Hercock, Eds., Springer, pp. 66-79. 2006.
26. Lam, D.N., J. Ahn, and K. S. Barber. "Software Engineering Tools for Designing and Comprehending Agent Systems," Journal Special Issue on Agent-oriented Software Development Methodologies (AOSDM) in an International Journal of Multi-Agent and Grid Systems, IOS Press, Amsterdam, The Netherlands; pp. 1-17, 2006.
27. Lam, D.N. and K. S. Barber. "Automated Interpretation of Agent Behavior," Springer-Verlag Lecture Notes in Artificial Intelligence (LNAI): Agent-Oriented Information Systems III: 7th International Bi-Conference AOIS 05 Workshop, Editors: Manuel Kolp, Paolo Bresciani, Brian Hendersen-Sellers, and Michael Winikoff, pp. 1-15, December 2006.
28. C.E. Martin and K.S. Barber. "Adaptive Decision-Making Frameworks for Dynamic Multi-Agent Organizational Change," International Journal on Multi-Agent and Autonomous Agent Systems, Vol. 13, No. 3, ISSN: 1387-2532 (print) or 1573-7454 (online), Vol. 13, No. 3, pp. 391-428, Springer Netherlands, 2006.
29. D. N. Lam, J. Ahn, and K. S. Barber. "Software Engineering Tools for Designing and Comprehending Agent Systems," International Journal of Multiagent and Grid Systems, H. Zhu and R. Unland, Eds., IOS Press, Volume 2, Number 4, pp. 473-489, 2006.
30. Fullam, K., T. Klos, G. Muller, J. Sabater, Z. Topol, K. S. Barber, J. Rosenschein, and L. Vercouter. "The Agent Reputation and Trust (ART) Testbed Architecture," Artificial Intelligence Research and Development, B. López, J. Meléndez, P. Radeva, and J. Vitrià, Eds., Vol. 131 of Frontiers in Artificial Intelligence and Applications, pp. 389-396, IOS Press, 2006.
31. Park, J., K. Fullam, D. Han, and K. S. Barber. "Agent Technology for Coordinating UAV Target Tracking," Knowledge-Based Intelligent Information and Engineering Systems, R. Khosla, R. Howlett, and L. Jain, Eds, pp. 831-837, Springer, 2005.
32. Fullam, K. and K. S. Barber. "A Temporal Policy for Trusting Information," Trusting Agents for Trusting Electronic Societies, R. Falcone, K. S. Barber, J. Sabater, and M. Singh, Eds., pp. 75-94, Springer, 2005
33. Fullam, K., J. Sabater, and K. S. Barber. "A Design Foundation for a Trust-Modeling Experimental Testbed," Trusting Agents for Trusting Electronic Societies, R. Falcone, K. S. Barber, J. Sabater, and M. Singh, Eds., pp. 95-109. Springer, 2005.
34. Park, J., Karen Fullam, David Han, K. S. Barber. "Agent Technology for Coordinating UAV Target Tracking," Knowledge-Based Intelligent Information and Engineering Systems, Lecture Notes in Computer Science, p. 831-837, Springer-Verlag, 2005.
35. Park, J. and K.S. Barber. "Information Quality Assurance by Lazy Exploration of Information Sources Combination Space in Open Multi-Agent Systems," Journal of Universal Computer Science, Vol. 11, Issue 1, pp. 193-209, 2005.
36. Barber, K.S. and M. MacMahon. "Analyzing Application Characteristics Motivating Adaptive Organizational Capabilities within Multi-Agent Systems," Journal of Autonomous Agents and Multi-Agent Systems, Special Issue on Application Science, Kluwer Academic Publishers, 2005.
37. Barber, K.S. and J. Park. "Agent Belief Autonomy in Open Multi-Agent Systems," Agents and Computational Autonomy: Potential, Risks and Solutions, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2969, pp. 7-16, 2004.
38. Barber, K.S. and M. MacMahon. "Quantifying the Search Space for Multi-Agent System (MAS) Decision-Making Organizations," (**Invited**) Connection Science, Special Issue: Agent Autonomy and Groups, Vol. 14, No. 4, 2003.

39. Barber, K.S., G. Baker, T.J. Graser, and J.C. Holt. “Arcade: Early Dynamic Property Evaluation of Requirements Using Partitioned Software Architecture Models,” Requirements Engineering Journal, Special Issue on Model-based Requirements Engineering, editors: Peri Loucopoulos and Colin Potts, Springer-Verlag, London, Vol. 8, No. 4, pp. 222-235, 2003.
40. Barber, K.S., T.J. Graser, J.C. Holt. “Evaluating Dynamic Correctness Properties of Domain Reference Architectures,” (**Invited**) Journal of Systems and Software, Vol. 68, Issue 3, pp. 217-231, December 2003.
41. Barber, K. S., A. Goel, D. C. Han, J. Kim, D. N. Lam, T. H. Liu, M. MacMahon, R. McKay, C. E. Martin. “Infrastructure for Design, Deployment and Experimentation of Distributed Agent-based Systems: The Requirements, The Technologies and An Example,” Journal of Autonomous Agents and Multi-Agent Systems, Special Issue on Infrastructure and Requirements for Building Research Grade Multi-Agent Systems, editors: Thomas Wagner and Omer Rana, Kluwer Academic Publishers, Vol. 7, No. 1 and 2, pp. 49-69, July 2003.
42. Barber, K.S. and J. Holt. “Software Architecture Correctness,” (**Invited**) IEEE Software, Vol. 18, No. 6, pp. 64-65, November 2002.
43. Barber, K. S. and C. E. Martin. “Dynamic Adaptive Autonomy in Multi-Agent Systems: Representation and Justification,” (**Invited**) International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), Special Issue on Intelligent Agent Technology, World Scientific Publishing Company, Vol. 15, No. 3, pp.405-433, 2001.
44. Barber, K. S. and Cheryl E. Martin. “Flexible Problem-Solving Roles for Autonomous Agents,” (invited) Integrated Journal on Computer-aided Engineering, Agent-Based Manufacturing Special Issue, Vol. 8, pp. 1–15, 2001.
45. Barber, K. S., T. H. Liu, S. Ramaswamy. “Conflict Detection During Plan-Integration for Multi-Agent Systems,” IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, Vol. 31, No. 4, pp. 616-628, 2001.
46. Barber, K. S., P. S. Grisham, S. R. Jernigan, T. J. Graser. “Requirements Evolution and Reuse Using the Systems Engineering Process Activities (SEPA),” Australian Journal of Information Systems (AJIS) Special Issue on Requirements Engineering, Vol. 7, No. 1, pp. 75–97, 2001.
47. Barber, K. S., R. McKay, A. Goel, D. Han, J. Kim, T. H. Liu, C. E. Martin. “Sensible Agents: The Distributed Architecture and Testbed,” IEEE and IEICE (The Institute of Electronics, Information and Communication Engineers) Joint Special Issue on Autonomous Decentralized Systems. Vol. E83-B, No. 5, pp. 951–960, 2000.
48. Barber, K. S. and J. Kim. “Toward Flexible and Fault Tolerant Intelligent Manufacturing: Sensible Agents in Shop-Floor Control,” (invited) Journal of Artificial Intelligence in Engineering Design and Manufacturing, Vol. 14, pp. 337–354, 2000.
49. Barber, K. S., A. Goel, C. E. Martin. “Dynamic Adaptive Autonomy in Multi-Agent Systems,” Journal of Experimental & Theoretical Artificial Intelligence, Special Issue on Autonomy Control Software, Publisher: Taylor and Francis Ltd, London, Vol. 12, No. 2, pp. 129–147, April, 2000.
50. Barber, K. S., A. Goel, D. Han, J. Kim, T. H. Liu, C. E. Martin, R. M. McKay. “Simulation Testbed for Sensible Agent-based Systems in Dynamic and Uncertain Environments,” (Invited) Transactions: Quarterly Journal of the Society for Computer Simulation International, Special Issue on Modeling and Simulation of Manufacturing Systems, Guest Editors: Michael Marefat and John Olson, Guest Editors, Vol. 16, No. 4, pp. 186–203, December, 1999.
51. Barber, K. S., T. H. Liu, A. Goel, and S. Ramaswamy. “Flexible Reasoning Using Sensible Agent-based Systems: A Case Study in Job Flow Scheduling,” Special Issue of Production Planning and Control, Vol. 10, No. 7, pp. 606–615, 1999.
52. Ramaswamy, S. and K. S. Barber. “A Design Architecture for Modeling, Analysis and Design of Manufacturing Control Software,” Journal of Intelligent Control and Systems, Vol. 2, No. 4, pp. 485–510, 1999.
53. Swaminathan, A., S. A. Shaikh, and K. S. Barber. “Design of an Experience-based Assembly Sequence Planner for Mechanical Assemblies,” Robotica: International Journal of Information, Education, and Research in Robotics and Artificial Intelligence, Special Issue on Intelligent Robotic Assembly, Vol. 16, No. 3, pp.265–283. May–June 1998.
54. Jernigan, S. R., S. Ramaswamy, and K. S. Barber. “A Distributed Search and Simulation Method for Job Flow Scheduling Simulation,” Simulation, Vol. 68, No. 6, pp. 377–401, June 1997.
55. Ramaswamy, S., K. P. Valavanis, K. S. Barber, “Petri Net Extensions for the Development of MIMO Net Models of Automated Manufacturing Systems,” Journal of Manufacturing Systems, Vol. 16, No. 3, pp. 175–191, May–June 1997.
56. Barcio, B.T., S. Ramaswamy, K. S. Barber. “An Object-Oriented Modeling and Simulation Environment for Reactive Systems Development,” Citation for Excellence Highest Quality Rating, International Journal of Flexible Manufacturing Systems, Vol. 9, No. 1, pp. 51–80, January 1997.
57. Suraj, A., S. Ramaswamy, and K. S. Barber, “Extended State Charts for the Modeling and Specification of Manufacturing Control Software Systems,” International Journal of Computer Integrated Manufacturing Systems, Special issue on Design

and Implementation of Computer-Integrated Manufacturing Systems: Integration and Adaptability Issues, Vol. 10, No. 1–4, pp.160–171, 1997.

58. Barber, K. S. "A Knowledge-based System Enabling Application Perspectives for CAD/CAM Integration," International Journal of Expert Systems: Research and Applications, Special Issue on Knowledge-based Planning, JAI Press, Vol. 9, No. 3, pp. 409–444, 1996.
59. Swaminathan, A. and K. S. Barber. "An Experience-based Assembly Sequence Planner for Mechanical Assemblies," IEEE Transactions on Robotics and Automation, Special Issue on Assembly and Task Planning for Manufacturing, Vol. 12, No. 2, pp. 252–267, April 1996.
60. Barcio, B. T., S. Ramaswamy, R. Macfadzean, K. S. Barber, "Object-Oriented Analysis, Modeling and Simulation of a Notional Air Defense System," Simulation, Vol. 66, No. 1, pp. 5–21, January 1996.
61. Barber, K. S., O. R. Mitchell, and K. A. Harbison-Briggs, "An Object-based System Incorporating Representations and Reasoning Mechanisms to Plan Manufacturing Applications," Journal of Systems Engineering, Vol. 5, pp. 36–47, 1995.

Book

1. K.S. Barber, et.al., "Handbook of Identity Management and Security: A Multidisciplinary Approach," in-progress and presenting to publishers.

Book Chapters

1. Zaeem, R. N., and K. S. Barber, "Privacy, a Machine Learning Perspective," (**Invited**) 3rd edition of Springer Encyclopedia of Cryptography, Security, and Privacy. (to appear December 2020)
2. Zaeem, R. N., and K. S. Barber, "Digital Identity Theft and Fraud," (**Invited**) 3rd edition of Springer Encyclopedia of Cryptography, Security, and Privacy. (to appear December 2020)
3. DeAngelis, D., R. Nokhbeh Zaeem, and, K. S. Barber, "Finding and Motivating Trusted Expert Participation in Online Communities," (**Invited**) Applications of Agent Models and Architectures, IGI Global, 2018.
4. Graser, T.J. and K.S. Barber. "Systematic Derivation and Evaluation of Domain-Specific, Implementation-Independent Software Architecture," Advances in Management Information Systems (AMIS) Monograph Series volume entitled Information Systems Analysis and Design: Foundation, Methods and Practices. Series Editor: Vladimir Zwass, Publisher M.E. Sharpe, Inc., April 2009.
5. Ahn, J, D. DeAngelis, and K. S. Barber. "Teammate Selection Using Multi-dimensional Trust and Attitude Models," Trust in Agent Societies, Springer Lecture Notes in Computer Science on 11th International Workshop, TRUST 2008, Estoril, Portugal, May 12 -13, 2008, Falcone, R.; Barber, K.S.; Sabater-Mir, J.; Singh, M.P. (Eds.) Series: Subseries: Lecture Notes in Artificial Intelligence, Vol. 5396 (ISBN: 978-3-540-92802-7), pp. 1-24, 2008.
6. Fullam, K. and K. S. Barber. "Information Valuation Policies for Explainable Trustworthiness Assessment in E-Services," Trust in E-Services: Technologies, Practices and Challenges, R. Song, L. Korba, and G. Yee, Eds., Idea Group, Chapter VII, pp. 168-197, 2007.
7. Fullam, K., T. Klos, G. Muller, J. Sabater, K. S. Barber, and L. Vercouter. "The Agent Reputation and Trust (ART) Testbed," Trust Management, Lecture Notes on Computer Science, Eds: K. Stølen, W. Winsborough, F. Martinelli, and F. Massacci, Springer Berlin / Heidelberg, Volume 3986/2006, pp. 439-442, 2006.
8. Fullam, K. and K. S. Barber. "A Temporal Policy for Trusting Information," Trusting Agents for Trusting Electronic Societies, R. Falcone, K. S. Barber, J. Sabater, and M. Singh, (Eds.), Lecture Notes in Computer Science, Springer Verlag, ISBN: 978-3-540-28012-5, Vol. 3577, pp. 75-94, 2005.
9. Fullam, K., J. Sabater, and K. S. Barber. "Toward a Testbed for Trust and Reputation Models," Trusting Agents for Trusting Electronic Societies, R. Falcone, K. S. Barber, J. Sabater, and M. Singh, (Eds.), Lecture Notes in Computer Science, Springer Verlag, pp. 95-109, 2005.
10. Vanzin, Marcelo M. and K.S. Barber. "Decentralized Partner Finding in Multi-Agent Systems," (**Invited**) Coordination of Large Scale Multi-agent Systems, ISBN: 978-0-387-26193-5, pp. 75-98, 2006.
11. Han, D. C. and K. S. Barber. "Desire-space Analysis and Action Selection for Multiple, Dynamic Goals," Computational Logic in Multi-Agent Systems, Lecture Notes in Artificial Intelligence, Edited by Leite, João; Torroni, Paolo, Springer-Verlag, ISBN: 3-540-28060-X, Vol. 3487, pp. 249-264, 2005.
12. Lam, D. N. and K. S. Barber. "Debugging Agent Behavior in an Implemented Agent System," Programming Multi-Agent Systems, Bordini, R.H.; Dastani, M.; Dix, J.; Seghrouchni, A.E.F. (Eds), Lecture Notes in Computer Science, Springer-Verlag, Vol. 3346, pp. 104-125, Spring 2005.

13. Barber, K.S. and M. MacMahon. “Application Characteristics Motivating Adaptive Organizational Capabilities within Multi-Agent Systems,” An Application Science for Multi-Agent Systems, Edited by Thomas Wagner, Kluwer Academic Publishers, pp. 175-198, 2004.
14. Barber, K. S., K. Fullam, and J. Kim. “Challenges for Trust, Fraud, and Deception Research in Multi-agent Systems,” () Trust, Reputation, and Security: Theories and Practice, Lecture Notes in Artificial Intelligence, Edited by R. Falcone, K.S. Barber, L. Korba, and M. Singh, Springer Press, pp. 8-14, 2003.
15. Barber, K. S. and J. Kim. “Soft Security: Isolating Unreliable Agents from Society,” (**Invited**) Trust, Reputation, and Security: Theories and Practice, Lecture Notes in Artificial Intelligence, Edited by R. Falcone, K. S. Barber, L. Korba, and M. Singh, Springer Press, pp. 224-234, 2003.
16. Barber, K.S. and J. Kim. “Belief Revision Process based on Trust: Agent Evaluating Reputation of Information Sources,” Trust in Cyber-societies: Integrating the Human and Artificial Perspectives, Lecture Notes in Computer Science, Edited by R. Falcone, M. Singh, and Y.H. Tan, Springer Press, Vol. 2246, pp. 73-82, 2002.
17. Barber, K.S., C. E. Martin, and R. M. McKay. “A Communication Protocol Supporting Dynamic Autonomy Agreements,” Advances in Artificial Intelligence. PRICAI 2000 Workshop Reader, Lecture Notes in Computer Science, Edited by R. Kowalczyk, S.W. Loke, N.E. Reed, and G. Williams, Springer-Verlag Heidelberg press (ISSN: 0302-9743), Vol. 2112/2001, pp. 303-320, 2001.
18. Barber, K.S. and C. E. Martin. “Autonomy as Decision-Making Control,” Intelligent Agents VII: Agent Theories, Architectures and Languages, Editors: Cristiano Castelfranchi and Yves Lesperance, Lecture Notes in Artificial Intelligence: Sub-series of Lecture Notes in Computer Science, Edited by J.G. Carbonell and J. Siekmann, Springer Press, Vol. 1986, pp. 343-345, 2001.
19. Barber, K.S., C. E. Martin, N. Reed, D. Kortenkamp, and G. Dorais. “Dimensions of Adjustable Autonomy: Panel Discussion,” Advances in Artificial Intelligence. PRICAI 2000 Workshop Reader, Lecture Notes in Artificial Intelligence, ISBN: 978-3-540-42597-7, Berlin, pp. 353-361, 2001.
20. Barber, K. S., D. C. Han, T. H. Liu. “Strategy Selection-based Meta-level Reasoning for Multi-Agent Problem Solving,” Agent Oriented Software Engineering, Editors: Paolo Ciancarini and Michael J. Wooldridge, Lecture Notes in Computer Science, Springer Press, Vol. 1957, pp. 269-284 of a 321 page book, January 2001.
21. Barber, K. S. and C.E. Martin. “The Motivation for Dynamic Decision-Making Frameworks in Multi-Agent Systems,” Agent Engineering, World Scientific Publishing, pp. 59-92, 2001.
22. Barber, K. S., T. H. Liu, D. C. Han. “Agent-Oriented Design,” Multi-Agent System Engineering: Proceedings of the 9th European Workshop on Modeling Autonomous Agents in a Multi-Agent World, MAAMAW'99 Valencia, Spain, June/July 1999, Springer Press, pp. 28–40, 1999.
23. Computer Science and Telecommunications Board, Manufacturing Studies Board, and National Research Council, Information Technology for Manufacturing: A Research Agenda, National Academic Press, pp. 92–94 and pp. 132–134, 1995.
24. Jernigan, S. R., S. Ramaswamy, and K. S. Barber. “Evaluation of On-line Schedules by Distributed Simulation,” Balanced Automation Systems: Architectures and design methods, IFIP and Chapman & Hall, pp. 195–202, 1995.
25. Barber, K. S., J. C. M. Tiernan, K. A. Harbison-Briggs, and O. R. Mitchell. “Symbolic Representation and Planning for Robot Control Systems in Manufacturing,” Artificial Intelligence: Applications in Manufacturing, American Association for Artificial Intelligence (AAAI) and MIT press, pp. 293–326, 1992.

Refereed Conference Proceedings

1. Chang, K. C., R. N. Zaeem, and K. S. Barber, "Is Your Phone You? How Privacy Policies of Mobile Apps Allow the Use of Your Personally Identifiable Information”, Proceedings of The Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications, 2020.
2. Zaeem, R. N., and K. S. Barber, “Comparing Privacy Policies of Government Agencies and Companies,” International Conference on Agents and Artificial Intelligence (ICAART) 2021, February 4-6, 2021. (accepted)
3. Chang, K. C., R.N. Zaeem, and K.S. Barber, “A Framework for Estimating Privacy Risk Scores of Mobile Apps,” 23rd Information Security Conference (ISC) 2020, December 16-20, 2020. (accepted)
4. Zaeem, R. N., C. Li, and K. S. Barber, “On Sentiment of Online Fake News,” Proceedings of the 2020 Proceedings of Foundations of Open Source Intelligence and Security Informatics FOSINT-SI. December 7-10, 2020.
5. Zaeem, R.N., Anya S., Issa, A., Nimergood, J. Rogers, I., Shah, V., Srivastava, A., and Barber, K.S. “PrivacyCheck’s Machine Learning to Digest Privacy Policies: Competitor Analysis and Usage Patterns”, The 2020 IEEE/WIC/ACM

International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'20), December 14-17, 2020 (accepted).

6. Liau, D., R.N. Zaeem, and K.S. Barber, “A Survival Game Analysis to Common Personal Identity Protection Strategies,” Proceedings of The Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications, December 7-11, 2020.
7. Chang, K. C., R.N. Zaeem, and K.S. Barber, “A Framework for Estimating Privacy Risk Scores of Mobile Apps,” The Annual Computer Security Applications Conference (ACSAC) 2020, Austin, TX, USA, December 7-11, 2020. (accepted)
8. Zaeem, R.N., Anya S., Issa, A., Nimergood, J. Rogers, I., Shah, V., Srivastava, A., and Barber, K.S. “PrivacyCheck v2: A Tool that Recaps Privacy Policies for You”, 29th ACM International Conference on Information and Knowledge Management (CIKM2020), October 19-23, 2020. (accepted)
9. Zaeem, R. N., and K. S. Barber, “How Much Identity Management with Blockchain Would Have Saved Us? A Longitudinal Study of Identity Theft,” Proceedings of the 3rd Workshop on Blockchain and Smart Contract Technologies (BSCT 2020), Colorado Springs, USA, June 8-10, 2020.
10. Liau, D., R.N. Zaeem, and K.S. Barber, “An Evaluation Framework for Future Privacy Protection Systems: A Dynamic Identity Ecosystem Approach,” Proceedings of the International Conference on Agents and Artificial Intelligence, Valletta, Malta, pp. 136-143, February 22-24, 2020.
11. Chen, C., R. N. Zaeem, and K.S. Barber, “Statistical Analysis of Identity Risk of Exposure and Cost Using the Ecosystem of Identity Attributes,” Proceedings of the European Intelligence and Security Informatics Conference (EISIC), Oulu, Finland, pp. 32-39, November 26-27, 2019.
12. Rana, R., R.N. Zaeem, and K.S. Barber, “US-Centric vs. International Personally Identifiable Information: A Comparison Using the UT CID Identity Ecosystem,” Proceedings of the 2019 IEEE International Carnahan Conference on Security Technology (ICCST), Montreal, Quebec, Canada, pp. 1-5. October 22-25, 2019.
13. Rana, R., R.N. Zaeem, and K.S. Barber, “An Assessment of Blockchain Identity Solutions: Minimizing Risk and Liability of Authentication,” Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI 2019). Thessaloniki, Greece, pp. 26-33, October 14-17, 2019.
14. Liau, D., R.N. Zaeem, and K.S. Barber, “An Evaluation Framework for Future Privacy Protection Systems: A Dynamic Identity Ecosystem Approach,” Proceeding of the IEEE International Conference on Privacy, Security and Trust, Fredericton, NB, Canada, pp. 339-341, August 26-28, 2019.
15. Liau, D., R.N. Zaeem, and K.S. Barber, “An Evaluation Framework for Future Privacy Protection Systems: A Dynamic Identity Ecosystem Approach,” Proceeding of the IEEE International Conference on Privacy, Security and Trust, Fredericton, NB, Canada, pp. 339-341, August 26-28, 2019Zaeem, R. N., D. Liau, and K.S. Barber, “Predicting Disease Outbreaks Using Social Media: Finding Trustworthy Users,” Proceedings of the IEEE Future Technologies Conference, pp. 369-384, Springer, November 15-16, 2018
16. Chang, K. C., R.N. Zaeem, and K.S. Barber, “Enhancing and Evaluating Identity Privacy and Authentication Strength by Utilizing the Identity Ecosystem,” ACM Proceedings of the 2018 Workshop on Privacy in the Electronic Society (WPES),Toronto, ON, Canada, pp. 114-120, October 15, 2018.
17. Chang, K.C., R. Nokhbeh Zaeem, and, K. S. Barber, “Internet of Things: Securing the Identity by Analyzing Ecosystem Models of Devices and Organizations,” AAAI Spring Symposia, Palo Alto, CA, USA, March 26-28, 2018.
18. Lin, G., R. Nokhbeh Zaeem, H. Sun, and K. Suzanne Barber, “Trust filter for Disease Surveillance: Identity, IEEE International Conference on Intelligent Systems (IntelliSys), London, UK, pp.1059--1066, September 7-8, 2017.
19. Lacey, David, James Zaiss, and K. Suzanne Barber, “Understanding Victim-enabled Identity Theft: Perpetrator and Victim Perspectives,” IEEE International Privacy, Security and Trust Conference, Auckland, New Zealand, pp. 196 – 202, December 12-14, 2016.
20. Zaeem, R. Nokhbeh, S. Budalakoti, M. Rasheed, C. Bajaj, K. Suzanne Barber, “Predicting and Explaining Identity Risk, Exposure and Cost Using Ecosystem of Identity Attributes,” IEEE International Carnahan Conference on Security Technology (ICCST), Orlando, Florida, pp. 176—183, October 24-27, 2016.
21. Zaeem, R. Nokhbeh, M. Manoharan, and K.S. Barber, “Risk Kit App: Highlighting Vulnerable Identity Assets for Specific Age Groups,” European Intelligence and Security Informatics Conference (EISIC), Uppsala, Sweden, pp. 32-38, August 15-19, 2016.
22. Hallenbeck, P. and K. Suzanne Barber, “Modeling and Enforcing Realism Requirements for Simulations,” ITEC 2017, Rotterdam, Netherlands, May 16-18, 2017.
23. Barber, K.S., “Identity Workforce Education,” Global Identity Summit, Tampa, FL, September 19-22, 2016.

24. Barber, K.S., "Identity as an Asset," Global Identity Summit, Tampa, FL, September 19-22, 2016.
25. K. Suzanne Barber, "The Identity of the Insider Threat," accepted to Defense Innovation Summit, Austin Texas, November 29 - December 1, 2016.
26. Zaeem, R. Nokhbeh and K. Suzanne Barber, "Trust Filtering to find the Best Sources in Social Media," accepted to Defense Innovation Summit, Austin Texas, November 29 - December 1, 2016.
27. Zaiss, James and K. Suzanne Barber, "Identity Threat Assessment and Prediction," accepted to Defense Innovation Summit, Austin Texas, November 29 - December 1, 2016.
28. Lacey, David, James Zaiss, and K. Suzanne Barber, "Understanding Victim-enabled Identity Theft: Perpetrator and Victim Perspectives," accepted to IEEE International Privacy, Security and Trust Conference, Auckland, New Zealand, December 12-14, 2016.
29. Zaeem, R. Nokhbeh and K.S. Barber, "Risk Kit App: Highlighting Vulnerable Identity Assets for Specific Age Groups", accepted by European Intelligence and Security Informatics Conference EISIC 2016, Uppsala, Sweden, August 15-19, 2016.
30. Barber, K.S., "Learning from the Criminal: Processes and Patterns of Identity Crimes," accepted to Global Identity Summit, Tampa, FL, September 21-24, 2015.
31. Barber, K.S., "The Emerging Identity Workforce," accepted to Global Identity Summit, Tampa, FL, September 21-24, 2015.
32. Soeder, B., and K.S. Barber, "A Model for Calculating User-Identity Trustworthiness in Online Transactions," International Conference on Privacy, Security, and Trust, Izmir, Turkey, pp. 177-185, July 21-23, 2015.
33. Yang, Y., M. Manoharan, and K.S. Barber, "Modelling and Analysis of Identity Threat Behaviors Through Text Mining of Identity Theft Stories," IEEE Joint Intelligence and Security Informatics Conference (JISI), the Hague, the Netherlands, pp. 184-191, September 24-26, 2014.
34. Soeder, B. and K.S. Barber, "Trustworthiness of Identity Attributes," in *Proceedings of the ACM 7th International Conference on Security of Information and Networks*, Glasgow, Scotland, UK, Sept. 9-11, 2014, pp. 4-9.
35. Hallenbeck, P. and K.S. Barber, "Near Real Time Certainty in Requirements Engineering," In the Proceedings of the IEEE Software Technology Conference, Long Beach, CA, April 1-3, 2014.
36. Soeder, B. and K.S. Barber, "Towards a Metric for Identity Confidence using an Agent Approach," In the Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART), March 6-8, 2014, ESEO, Angers, Loire Valley, France.
37. Budalakoti, S. and K. S. Barber. "Tournament-based Reputation Models for Aggregating Relative Preferences," In the Proceedings of the ASE/IEEE International Conference on Social Computing (SocialCom 2013), September 8-14, 2013, Washington, DC.
38. Budalakoti, S. and K. S. Barber. "Reputation Models for Aggregating Relative Preferences," In the Proceedings of The Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2013), May 7, 2013, Minneapolis, MN.
39. Budalakoti, S. and K. S. Barber. "Reputation Estimation based on Online Social Network Structure: A Relational Capital Model," In the Proceedings of The Workshop on Trust in Agent Societies at The Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2012), June 5, 2012, Valencia Spain.
40. Budalakoti, S. and K. S. Barber. "The Potential of Online Social Network Data in Identity Management," In the Proceedings of ID360, April 23-24, 2012, Austin, TX.
41. Budalakoti, S., D. DeAngelis, and K. S. Barber. "Unbiased Trust Estimation in Content-Oriented Social Networks," In the Proceedings of The Workshop on Trust in Agent Societies at The Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2011), pp. 13-24, May 2, 2011, Taipei, Taiwan.
42. Budalakoti, S. and K. S. Barber. "Authority vs Affinity: Modeling User Intent in Expert Finding," In Proceedings of the International Symposium on Social Intelligence and Networking (SIN-10) at The 2010 IEEE Conference on Social Computing (SocialCom-10), August 20-22 2010, Minneapolis, MN.
43. Kadaba, R., S Budalakoti, D. DeAngelis, and K. S. Barber. "Modeling Virtual Footprints," In the Proceedings of The Workshop on Trust in Agent Societies at The Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2010), May 10-14, 2010, Toronto, Canada.
44. Budalakoti, S., D DeAngelis, and K. S. Barber. "Expertise Modeling and Recommendation in Online Question and Answer Forums," In the Proceedings of the International Symposium on Social Intelligence and Networking (SIN-09) at The 2009 IEEE Conference on Social Computing (SocialCom-09), August 29-31, 2009, Vancouver, Canada.

45. Budalakoti, S., D. DeAngelis, and K. S. Barber. "Expertise Modeling and Recommendation in Online Question and Answer Forums," In the Proceedings of the 12th IEEE International Conference on Computer Science and Engineering (CSE-09), Vol. 4, pp. 481 – 488, August 29-31, 2009, Vancouver, Canada.
46. Jones, C.E., and K.S Barber. "Dross into Diamonds: Efficient Use of Untrustworthy Agents in a Large Scale Environment," In Proceedings of the Third International Workshop on Massively Multi-Agent Systems: Models, Methods and Tools (MMAS'09), May 12, 2009, Budapest, Hungary.
47. Ahn, J., X. Sui, D. DeAngelis, and K. S. Barber. "Identifying Beneficial Teammates using Multi-Dimensional Trust," In Proceedings of Seventh International Joint Conference on Autonomous Agents and Multi-Agent System, pp. 1469-1472, May 12-16, 2008, Estoril, Portugal.
48. Jones, C.L.D., and K. S. Barber. "Combining Job and Team Heuristics," Workshop on Coordination, Organization, Institutions, and Norms; In Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent System, May 12, 2008, Estoril, Portugal.
49. Budalakoti, S., D. DeAngelis, and K.S. Barber. "Expertise Modeling and Recommendation in Online Question and Answer Forums," In Proceedings of the 2009 IEEE International Conference on Social Computing, Symposium on Social Intelligence and Networking (SIN-09), pp.481-488, August 29-31, 2009, Vancouver, Canada.
50. Fullam, K. and K.S. Barber. "Dynamically Learning Sources of Trust Information: Experience vs. Reputation," In Proceedings of The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2007), ISBN:978-81-904262-7-5, Article 164, May 14-18, 2007, Honolulu, HI.
51. DeAngelis, D. and K. S. Barber. "Applying Trust to a Secure Message Passing Domain," In Proceedings of The Workshop on Trust in Agent Societies at The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2007), pp. 39-43, May 14-18, 2007, Honolulu, HI.
52. Budalakoti, S., Daiqian Zhan, K. S. Barber. "Reputation-based Security for Agent Communities in a Rapidly Evolving Environment," In Proceedings of Workshop on Trust in Agent Societies at The Seventh International Joint Conference on Autonomous Agents and Multi-agent Systems, May 14-18, 2007, Honolulu, Hawaii.
53. Barber, K.S., J. Ahn, S. Budalakoti, D. DeAngelis, K. K. Fullam, C. L.D. Jones, and X. Sui. "Agent Trust Evaluation and Team Formation in Heterogeneous Organizations," In Proceedings of The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2007) Demonstration Track, May 14-18, 2007, Honolulu, HI.
54. Barber, K.S. and C.L.D. Jones. "Bottom-up Team Formation in a Dynamic Environment," In Proceedings of the CCMMS 2007 Workshop at The Seventh International Joint Conference on Autonomous Agents and Multi-agent Systems, May 14-18, 2007, Honolulu, HI.
55. Ahn, J., D. DeAngelis, and K. S. Barber. "Attitude Driven Team Formation using Multi-Dimensional Trust," In Proceedings of the 2007 IEEE WIC ACM International Conference on Intelligent Agent Technology, November 2-5, 2007, San Jose, CA.
56. Ahn, J., C. L. D. Jones, and K. S. Barber. "Identifying Optimal Jobs to Work On: The Role of Attitude in Job Selection," In Proceedings of the 2007 IEEE / WIC / ACM International Conference on Intelligent Agent Technology, Nov 2-5, 2007, Silicon Valley, CA.
57. Jones, C.L.D. and K. S. Barber. "Exploiting Untrustworthy Agents in Team Formation," In Proceedings of the 2007 IEEE / WIC / ACM International Conference on Intelligent Agent Technology, Nov 2-5, 2007, Silicon Valley, CA.
58. Bosse, T., D. N. Lam, and K. S. Barber. "Automated Analysis and Verification of Agent Behavior," In Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems, pp. 1317-1319, May 8-12, 2006, Hakodate, Japan.
59. Gujral, N., D. DeAngelis, K. K. Fullam and K. S. Barber. "Modeling Multi-Dimensional Trust," In Proceedings of the Workshop on Trust in Agent Societies at The 5th International Conference on Autonomous Agents and Multiagent Systems, pp. 35-41, May 8-12, 2006, Hakodate, Japan.
60. Bosse, T., D. N. Lam, and K. S. Barber. "Empirical Analysis for Agent System Comprehension and Verification," In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006), December 2006, Hong Kong.
61. Park, J. and K.S. Barber. "Collaboration among Competitve Agents in Information Sharing Networks," In Proceedings of the 2nd International Workshop on Coordination and Organisation (CoOrg 2006), in conjunction with Distributed Computing Techniques (DisCoTec06), June 13, 2006, Bologna, Italy.
62. Fullam, K., T. Klos, G. Muller, J. Sabater, K. S. Barber, and L. Vercouter. "The Agent Reputation and Trust (ART) Testbed," In Proceedings of The BeNeLux? Conference on Artificial Intelligence (BNAIC 2006), pp. 449-450, October 5-6, 2006, Namur, Belgium.

63. Fullam, K. and K.S. Barber. "Learning Trust Strategies in Reputation Exchange Networks," In Proceedings of The Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2006), pp. 1241-1248, May 8-12, 2006, Hakodate, Japan.

64. Han, D., J. Park, K. Fullam, and K. S. Barber. "Application of Action Selection, Information Gathering, and Information Evaluation Technologies to UAV Target Tracking," In Proceedings of the Defence Applications of Multi-Agent Systems: International Workshop, DAMAS 2005, Invited Papers, S. Thompson and R. Ghanea-Hercock, Eds., Springer, pp. 66-79, July 25, 2005, Utrecht, The Netherlands.

65. Lam, D. N. and K. S. Barber. "Automated Interpretation of Agent Behavior," In Proceedings of the Workshop for Agent-Oriented Information Systems (AOIS-2005) at International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 74-81, July 26, 2005, Utrecht, Netherlands.

66. Park, J., J. Ahn, D. DeAngelis, K. K. Fullam, N. Gujral, D. C. Han, D. N. Lam, and K. S. Barber. "Design, Runtime, and Analysis of Multi-Agent Systems," In Proceedings of the 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2005), pp. 157-158, July 26, 2005, Utrecht, Netherlands.

67. Fullam, K., T. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. Rosenschein, L. Vercouter, and M. Voss. "A Specification of the Agent Reputation and Trust (ART) Testbed: Experimentation and Competition for Trust in Agent Societies," In Proceedings of The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005), pp. 512-518, July 25-29,2005, Utrecht, Netherlands.

68. Fullam, K., T. Klos, G. Muller, J. Sabater, Z. Topol, K. S. Barber, J. Rosenschein, and L. Vercouter. "Le banc d'essais ART (Agent Reputation and Trust) pour les modèles de confiance," In Proceedings of the Actes des Journées Francophones sur les Systèmes Multi-Agents, A. Drogoul and É. Ramat, Eds., pp. 175-179, Hermès: Calais, France.

69. Barber, K.S., J. Ahn, D. DeAngelis, K. Fullam, N. Gujral, D. Han, D. Lam, and J. Park. "Design, Runtime, and Analysis of Multi-Agent Systems," In Proceedings of The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005) Industry Track, pp. 157-158, July 25-29, 2005, Utrecht, Netherlands.

70. Fullam, K., J. Park, and K. S. Barber. "Trust-driven Information Acquisition for Secure and Quality Decision-Making," In Proceedings of The International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS-2005), pp. 303-310, April 18-21, 2005, Waltham, MA.

71. Fullam, K., T. Klos, G. Muller, J. Sabater, Z. Topol, K. S. Barber, J. Rosenschein, and L. Vercouter. "The Agent Reputation and Trust (ART) Testbed Architecture," In Proceedings of the Workshop on Trust in Agent Societies at The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005), pp. 50-62, July 25-29, 2005, Utrecht, The Netherlands.

72. DeAngelis, D., K. Fullam, and K. S. Barber. "Effects of Communication Disruption in Mobile Agent Trust Assessments for Distributed Security," In the Proceedings of the Workshop on Trust in Agent Societies at The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005), pp. 27-37, July 25-29, 2005, Utrecht, The Netherlands.

73. Park, J. and K. S. Barber. "Fault-Tolerant Information Sharing Networks," In Proceedings of the 2nd International Workshop on Safety and Security in Multi-Agent Systems (SASEMAS 2005), 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), pp. 76-88, July 25-29, 2005, Utrecht, The Netherlands.

74. Han, D., J. Park, K. Fullam, and K. S. Barber. "Application of Action Selection, Information Gathering, and Information Evaluation Technologies to UAV Target Tracking," In Proceedings of the Workshop on Defense Applications of Multi-Agent Systems (DAMAS 05), 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), pp. 43-54, July 25-29, 2005, Utrecht, The Netherlands.

75. Fullam, K., T. Klos, G. Muller, J. Sabater, Z. Topol, K. S. Barber, J. Rosenschein, and L. Vercouter. "A Demonstration of The Agent Reputation and Trust (ART) Testbed: Experimentation and Competition for Trust in Agent Societies," In the Proceedings of The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005) Industry Track, pp. 151-152, July 25-29, 2005, Utrecht, The Netherlands.

76. Barber, K.S., J. Ahn, D. DeAngelis, K. Fullam, N. Gujral, D. C. Han, D. N. Lam, J. Park. "Design, Runtime, and Analysis of Multi-Agent Systems," In Proceedings of the 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), pp. 157-158, July 25-29, 2005, Utrecht, The Netherlands.

77. Gujral, N., J. Ahn, and K.S. Barber. "Architectural Model for Designing Agent-based System," In Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering, pp. 753-760, July 14-15, 2005, Taipei, Taiwan, Republic of China.

78. Fullam, K., T. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. Rosenschein, L. Vercouter, and M. Voss. “A Competition Testbed for Trust in Agent Societies,” In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005), pp. 512-518, July 25-29, 2005, Utrecht, Netherlands.

79. Lam, D. N. and K.S. Barber. "Comprehending Agent Software,” In Proceedings of The International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), pp. 586-593, July 25-29, 2005, Utrecht, Netherlands.

80. Lam, D. N. and K. S. Barber. "Automated Interpretation of Agent Behavior,” In Proceedings of the Workshop for Agent-Oriented Information Systems (AOIS-2005) at The International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), pp. 74-81, July 26, 2005, Utrecht, Netherlands.

81. Han, D. C. and K. S. Barber. “Determining Task Valuations for Task Allocation,” In Proceedings of the International Conference on Automated Planning & Scheduling, Workshop on Multi-agent Planning & Scheduling, AAAI Press, pp. 73-79, June 5-10, 2005, Monterey, CA.

82. Han, D. C. and K. S. Barber. “Responding to Uncertainty in UAV Surveillance through Desire Analysis,” In Proceedings of the International Conference on Automated Planning & Scheduling, Workshop on Planning under Uncertainty for Autonomous Systems, AAAI Press, pp. 49-56, June 5-10, 2005, Monterey, CA.

83. Fullam, K., J. Park, and K. S. Barber. “Trust-driven Information Acquisition for Secure and Quality Decision-Making,” Proceedings of The International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS-2005), pp. 303-310, April 18-21, 2005, Waltham, MA.

84. Ahn, J., T. Graser, D. Lam, and K.S. Barber. "System Engineering Processes Activities for Agent System Design: Component-based Development for Rapid Prototyping,” In Proceedings of International Conference on Enterprise Information Systems, pp. 196-202, May 24-28, 2005, Miami, FL.

85. Fullam, K. and K. S. Barber. “Evaluating Approaches for Trust and Reputation Research: Exploring a Competition Testbed,” In Proceedings of the Workshop on Reputation in Agent Societies, Intelligent Agent Technology (IAT-2004), pp. 1-6, September 20-23, 2004, Beijing, China.

86. Han, D.C. and K. S. Barber. “Desire-space Analysis and Action Selection for Multiple, Dynamic Goals,” In Proceedings of CLIMA V, Fifth Workshop on Computational Logic in Multi-agent Systems, pp 182-195, September 29-30, 2004, Lisbon, Portugal.

87. Fullam, K. and K. S. Barber. “A Temporal Policy for Trusting Information,” In Proceedings of the Workshop on Trust in Agent Societies, The Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS2004), pp. 47-57, July 19-23, 2004, New York City, NY.

88. Barber, K.S., N. Gujral, D. N. Lam, J. Ahn, and T. J. Graser. “Agent Technology Portfolio Manager,” In Proceedings of the 16th International Conference on Software Engineering and Knowledge Engineering, pp. 37-44, June 20-24, 2004, Banff, Canada.

89. Fullam, K. and K. S. Barber. “Using Policies for Information Valuation to Justify Beliefs,” In Proceedings of The Third International Joint Conference on Autonomous Agents and Multi Agent Systems, pp. 404-411, July 19-23, 2004, New York City, NY.

90. Lam, D.N. and K. S. Barber. “Verifying and Explaining Agent Behavior in an Implemented Agent System," In Proceedings of The Third International Joint Conference on Autonomous Agents and Multi Agent Systems, pp. 2336 – 7, July 19-23, 2004, New York City, NY.

91. Lam, D. N. and K. S. Barber. “Debugging Agent Behavior in an Implemented Agent System," In Proceedings of The Third International Joint Conference on Autonomous Agents and Multi Agent Systems, Workshop on Programming Multi-Agent Systems Languages and Tools, pp. 45-56, July 20, 2004, New York City, NY.

92. Vanzin, M. and K. S. Barber. “Decentralized Partner Finding in Multi-Agent Systems,” In Proceedings of The Third International Joint Conference on Autonomous Agents and Multi Agent Systems, Workshop on Challenges in the Coordination of Large Scale Multi-Agent Systems, July 20, 2004, New York City, NY.

93. Park, J. and K. S. Barber. “Finding Information Sources by Model Sharing in Open Multi-Agent Systems,” In Proceedings of the Workshop on Agents for Ubiquitous Computing (UbiAgents), The Third International Joint Conference on Autonomous Agents and Multi Agent Systems, pp. 21-28, July 20, 2004, New York City, NY.

94. Barber, K.S., J. Park, R. McKay, J. Ahn, K. Fullam, D. N. Lam, M. M. Vanzin, T. J. Graser, D. C. Han, and N. Gujral. “Multi-agent System Development: Design, Runtime, and Analysis,” In Proceedings of the 19th National Conference on AI (AAAI-2004) Intelligent Systems Demonstration. pp. 1006-1007, July 25-29, 2004, San Jose, CA.

95. Barber, K.S. and J. Park. “Robust Partner Selection Scheme for Information Quality Assurance despite Uncertainty in Open Multi-Agent Systems,” In Proceedings of the IEEE International Conference on Information Technology (ITCC04), Vol. 1, pp.430-434, April 5-7, 2004, Las Vegas, NV.

96. Barber, K. S. and J. Park. "Finding Partners to Form Information Sharing Networks in Open Multi-Agent Systems," In Proceedings of Florida Artificial Intelligence Research Society (FLAIRS 2004), pp. 412-417, May 14-20, 2004, Miami, FL

97. Kasputis, S., I. Oswalt, R. McKay, and K. S. Barber. "Semantic Descriptors of Models and Simulations," In Proceedings of the Simulation Interoperability Workshop, April 18-23, 2004, http://www.sisostds.org/, Arlington, VA.

98. Barber, K. S., K. Fullam, T.J. Graser, D.C. Han, J. Kim, D.N. Lam, R.M. McKay, J. Park, and M. Vanzin. "Distributed Biosurveillance Systems using Sensible Agent Technology to Improve Coordination and Communication among Decision-makers," In Proceedings of the 2003 Joint Scientific Conference on CB Defense, p. 40, November 17-20, 2003, Baltimore, MD.

99. Barber, K. S., D. Faith, K. Fullam, T. Graser, D. C. Han, J. Jeong, J. Kim, D. Lam, R. McKay, M. Pal, J. Park, and M. Vanzin. "Sensible Agent Technology Improving Coordination and Communication in Biosurveillance Domains," In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 1631-1632, August 9-15, 2003, Acapulco, Mexico.

100. Barber, K. S. and D. N. Lam. "Enabling Abductive Reasoning for Agent Software Comprehension" In Proceedings of the Agents and Automated Reasoning Workshop, Eighteenth International Joint Conference on Artificial Intelligence, pp. 7-13, August 11, 2003, Acapulco, Mexico.

101. Barber, K. S. and D. N. Lam. "Motivating Abductive Explanation for Multi-Agent System Comprehension," (**Invited**) In Proceedings of the 7th World Multi-conference on Systemics, Cybernetics and Informatics (SCI 2003), Session on Scientific and Mathematical Foundations of Agent-Based Computing, pp. 484-489, July 27-30, 2003, Orlando, FL.

102. Barber, K. S. and J. Park. "Autonomy Affected by Beliefs: Building Information Sharing Networks with Trustworthy Providers," In Proceedings of the Workshop on Autonomy, Delegation, and Control: From Inter-agent to Organizations and Institutions, The Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2003), pp. 11-16, July 14–18, 2003, Melbourne, Australia.

103. Barber, K. S., T.J. Graser, and M.M. Vanzin. "Challenges in Dynamic Coalition Formation Observations from the Coalition Agents eXperiments," In Proceedings of the Workshop on Representation and Approaches for Time-Critical Decentralized Resource/Role/Task Allocation Workshop, The Second International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2003), pp. 7-13, July 14–18, 2003, Melbourne, Australia.

104. Barber, K. S., D. C. Han, K. Fullam, J. Jeong, J. Kim, J, Park, and M, Vanzin. "Identically Handling Interactions with Human and Software Agents," (**Invited**) In Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2003), Workshop on Humans and Multi-Agent Systems, pp. 36-42, July 14–18, 2003, Melbourne, Australia.

105. Barber, K. S. and K. Fullam. "Applying Reputation Models to Continuous Belief Revision," Workshop on Deception, Fraud and Trust in Agent Societies, In Proceedings of The Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2003), pp. 6-15, July 14–18, 2003, Melbourne, Australia.

106. Barber, K. S. and D. N. Lam. "Specifying and Analyzing Agent Architectures using the Agent Competency Framework," In Proceedings of the 15th International Conference in Software Engineering and Knowledge Engineering, pp. 232-239, July 1-3, 2003, Redwood City, CA.

107. Barber, K.S. and J. Kim. "Soft Security: Isolating Unreliable Agents from Society," (**Invited**) In Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002), 5th Workshop on Deception, Fraud and Trust in Agent Societies, pp. 8-17, July 15-19, 2002, Bologna, Italy.

108. Barber, K.S. and M. MacMahon. "Analyzing Factors Influencing Decision making Frameworks and Impacting Agent and Multi-Agent System Performance," (**Invited**) In Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002), Workshop on "Toward an Application Science: MAS Problem Spaces and Their Implications to Achieving Globally Coherent Behavior," pp. 20-27, July 15-19, 2002, Bologna, Italy.

109. Barber, K.S. and M. MacMahon. "Quantifying the Search Space for Multi-Agent Systems (MAS) Decision Making Organizations," In Proceedings of the18th National Conference on Artificial Intelligence (AAAI 2002), Workshop on Autonomy, Delegation, and control: From Inter-agent to Groups, pp. 1-7, July 28 – August 1, 2002, Edmonton, Alberta, Canada.

110. Barber, K.S. and M. MacMahon. "Challenges in Identifying the Best Agent Organizations," In Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002), Teamwork and Coalition Formation Workshop, pp. 45-52, July 15-19, 2002, Bologna, Italy.

111. Barber, K. S. and D. N. Lam. "Architecting Agents Using Core Competencies," In Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems, pp. 90-91, July 15-19, 2002, Palazzo Re Enzo, Bologna, Italy.

112. Barber, K.S., J. Holt and G. Baker. "Early Multi-Level Software Architecture Performance Evaluations," In Proceedings of the 15th International Conference on Software & Systems Engineering and their Applications (ICSSEA 2002), December 3 -5, 2002, Paris, France.

113. Barber, K.S., T. Graser and J. Holt. "Enabling Iterative Software Architecture Derivation Using Early Non-Functional Property Evaluation," In Proceedings of the IEEE International Conference on Automated Software Engineering (ASE2002), September 23-27, 2002, Edinburgh, UK.

114. Barber, K.S., G. Baker and J. Holt. "Performance Evaluation of Domain Reference Architectures," In Proceedings of the 14th International Conference in Software Engineering and Knowledge Engineering (SEKE 2002), pp. 225-232, July 15-18, 2002, Ischia, Italy.

115. Barber, K.S. and J. Holt. "Supporting Separation of Concerns During Software Architecture Performance Evaluations," In Proceedings of the 6th World Multi-conference on Systemics, Cybernetics and Informatics (SCI2002), pp. 333-338, July 14-18, 2002, Orlando, FL.

116. Barber, K.S. and C. Martin. "Autonomy of Decision-Makers in Coalitions," In Proceedings of the Second International Conference on Knowledge Systems for Coalition Operations (KSCO-2002), pp. 22-24, April 23-24, 2002, Toulouse, France.

117. Barber, K.S., T. Graser and J. Holt. "Evolution of Requirements and Architectures: An Empirical-based Analysis," In Proceedings of the First International Workshop on Model-based Requirements Engineering (MBRE'01), pp. 9-16, November 30, 2001, San Diego, CA.

118. Barber, K. S., A. Goel, J. Kim, C.E. Martin, D.C. Han, D.N. Lam, M. MacMahon, and R. McKay, "Sensible Agents: Augmenting and Empowering Human Decision Makers," In Proceedings of the Annual Conference of the European Chapter of Human Factors and Ergonomics Society, pp. 1-14, November 7-9, 2001, Torino, Italy.

119. Barber, K.S., T. Graser, and K. Henry. "Advanced Consequence Management Program: Challenges and Recent Real World Implementations," In Proceedings of the SPIE's 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls - Track on Enabling Technologies for Law Enforcement and Security," Proceedings on CD-ROM, April 1-5, 2002, Orlando, FL.

120. Barber, K.S., T. Graser and J. Holt. "Providing Early Feedback in the Development Cycle through Automated Application of Model Checking to Software Architectures," In Proceedings of the IEEE International Conference on Automated Software Engineering Conference (ASE2001), pp. 341-345, November 26-29, 2001, San Diego, CA.

121. Barber, K.S., T. Graser, A. Bingham and J. Holt. "Reliability Estimation Techniques for Domain Reference Architectures," In Proceedings of the 14th International Conference on Software & Systems Engineering and their Applications (ICSSEA 2001), Vol. 3, section 12-4, pp. 1-8, December 4-6, 2001, Paris, France.

122. Barber, K.S. and A. Lakshmanan. "Analysis of Software Architectures to Generate Test Sequences," In Proceedings of the 14th International Conference on Software & Systems Engineering and their Applications (ICSSEA 2001), Vol. 3, section 11-5, pp.1-8, December 4-6, 2001, Paris, France.

123. Barber, K. S., I.M. Gamba, and C.E. Martin. "Analysis of Adaptive Decision-Making Frameworks: Motivation for Adjusting Autonomy through Decision-Making Control," In Proceedings of the IJCAI-01 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents, pp. 9-13, August 4-10, 2001, Seattle, WA.

124. Barber, K. S., D.N. Lam and R. McKay. "Application of Sensible Agents Supporting Electronic Commerce within the Supply Chain," In Proceedings of the 17th International Joint Conference on Artificial Intelligence, Workshop on Artificial Intelligence and Manufacturing: New AI Paradigms for Manufacturing, pp. 9-15, August 4-10, 2001, Seattle, WA.

125. Barber, K. S., T. Graser, S. Bhattacharya, and J. Holt. "A Multi-Level Software Architecture Metamodel to Support the Capture and Evaluation of Stakeholder Concerns," In Proceedings of the 5th World Multi-conference on Systematics, Cybernetics and Informatics (SCI2000), **Best Paper Award**, pp. 337-342, July 22–25, 2001, Orlando, FL.

126. Barber, K. S., T. Graser, and J. Holt. "Evaluating Dynamic Correctness Properties of Domain Reference Architectures Using a Combination of Simulation and Model Checking," In Proceedings of the International Conference on Software Engineering & Knowledge Engineering (SEKE2001), **Best Paper Award**, pp. 19-28, June 12–14, 2001, Buenos Aires, Argentina.

127. Barber, K. S., I.M. Gamba, and C.E. Martin. "Analysis of Adaptive Decision-Making Frameworks," (**Invited**) In Proceedings of the Workshop on Autonomy Oriented Computation (AOC), 5th International Conference on Autonomous Agents, pp. 12-20, May 28-June 1, 2001, Montreal, Canada.

128. Barber, K. S. and J. Kim. "Belief Revision Process based on Trust: Simulation Experiments," (**Invited**) In Proceedings of the Workshop for Fraud, Deception and Trust in Agent Societies, Autonomous Agents 2001, pp. 1-12, May 28-June 1, 2001, Montreal, Canada.

129. Barber, K. S. and C.E. Martin. "Dynamic Reorganization of Decision-Making Groups," In Proceedings of the Autonomous Agents 2001, pp. 513-520, May 28-June 1, 2001, Montreal, Canada.

130. Barber, K. S., R. McKay, M. MacMahon, C.E. Martin, D.N. Lam, A. Goel, D.C. Han, and J. Kim, "Sensible Agents: An Implemented Multi-Agent System and Testbed," In Proceedings of The 5th International Conference on Autonomous Agents 2001: Implemented Systems Track, pp. 92-99, May 28-June 1, 2001, Montreal, Canada.

131. Barber, K. S., M. MacMahon, R. McKay, A. Goel, D.C. Han, J. Kim, D.N. Lam, and C.E. Martin. "An Agent Infrastructure Implementation for Leveraging and Collaboration in Operational and Experimental Environments," (**Invited**) In Proceedings of the 5th International Conference on Autonomous Agents, Workshop on Infrastructure for Agents, MAS, and Scalable MAS, pp. 41-46, May 28-June 1, 2001, Montreal, Canada.

132. Barber, K. S., T.J. Graser, and Col. John Silva, M.D. "Developing a Traceable Domain Reference Architecture to Support Clinical Trials at the National Cancer Institute – An Experience Report," In Proceedings of the 8th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2001), pp. 144-151, April 18–19, 2001, Washington, D.C.

133. Barber, K. S. and S. Bhattacharya. "A Representational Framework for Technology Component Reuse," In Proceedings of the 13th International Conference on Software & Systems Engineering and their Applications (ICSSEA'2000), December 5–8, 2000 Paris, France.

134. Barber, K. S. and T. Graser. "Tool Support for Systematic Class Identification in Object-Oriented Software Architectures," In Proceedings of the 37th International Conference on Technology of Object-Oriented languages and Systems (TOOLS-37), pp. 82-93, November 20–23, 2000, Sydney, Australia.

135. Barber, K. S. and S. Bhattacharya. "Representing Technology to Promote Reuse in the Software Design Process," In Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE 2000), No. 11–15, pp. 285–288, September 11–15, 2000, Grenoble, France.

136. Barber, K. S., D. C. Han, and T. H. Liu. "Coordinating Distributed Decision Making Using Reusable Interaction Specifications," In Proceedings of the 3rd Pacific Rim International Workshop on Multi-Agents (PRIMA 2000), Zhang, C. and Soo, V.W., Eds. New York: Springer, pp. 1 – 15, August 28–29, 2000, Melbourne, Australia.

137. Barber, K. S., C. E. Martin and R. McKay. "A Communication Protocol Supporting Dynamic Autonomy Agreements," (**Invited**) In Proceedings of the Sixth Pacific Rim International Conference on Artificial Intelligence (PRICAI), Workshop on Teams with Adjustable Autonomy, pp. 1 –10, August 28–29, 2000, Melbourne, Australia.

138. Barber, K. S. "Sensible Agents Capable of Dynamic Adaptive Autonomy: An Architecture and Infrastructure to Support Reconfiguration of Problem Solving Frameworks for Electronic Commerce Applications," (**Invited**) In Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet," CD ROM, August 5, 2000, L'Aquila, Italy.

139. Barber, K. S., A. Goel, D. C. Han, J. Kim, D. N. Lam, T. H. Liu, C. E. Martin, and R. McKay. "Sensible Agents: Demonstration of Dynamic Adaptive Autonomy," (**Invited**) In Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 1115-1116, July 30 – August 3, 2000, Austin, TX.

140. Barber, K. S., T. Graser, P. Grisham, S. Jernigan, and S. Bhattacharya. "The Systems Engineering Process Activities (SEPA) Methodology and Tool Suite," (**Invited**) In Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 1117–1118, July 30 – August 3, 2000, Austin, TX.

141. Barber, K. S. and D. N. Lam. "Tracing Dependencies of Strategy Selections in Agent Design," In Proceedings of the 17th National Conference on Artificial Intelligence, p. 1082, July 30-August 3, 2000, Austin, TX.

142. Barber, K. S. and T. Graser. "Reference Architecture Representation Environment (RARE) – A Tool to Support Object-Oriented Software Architecture Derivation and Evaluation," In Proceedings of the 4th World Multi-conference on Systematics, Cybernetics and Informatics (SCI2000), Vol. 2, pp. 389–394, July 23–26, 2000, Orlando, FL.

143. Barber, K .S. and Joon Kim. "Constructing and Dynamically Maintaining Perspective-based Agent Models for Command and Control Applications in a Multi-Agent Environment," In Proceedings of the Sixth International Conference on Intelligent Autonomous, pp. 725–735, July 25–27, 2000, Venice, Italy.

144. Barber, K. S., T. Graser, J. Holt, and J. Silva. "Representing Domain Reference Architectures by Extending the UML Metamodel," In Proceedings of the Twelfth International Conference on Software Engineering (SEKE 2000), pp. 256–265, July 6–8, 2000, Chicago, IL.

145. Barber, K. S. and Joon Kim. "Belief Revision Process based on Trust: Agents Evaluating Reputation of Information Sources," In Proceedings of the International Conference on Autonomous Agents 2000 Workshop on Deception, Fraud, and Trust in Agent Societies, pp. 15–26, June 2–6, 2000, Barcelona, Spain.

146. Barber, K. S. and T. H. Liu. “Conflict Detection During Plan Integration based on the Extended PERT Diagram,” In Proceedings of the International Conference on Autonomous Agents (Agents 2000), ACM, Inc. pp. 106–107, June 3–7, 2000, Barcelona, Catalonia, Spain.

147. Barber, K. S., D. N. Lam, C. E. Martin, and R. M. McKay. “Sensible Agent Testbed Infrastructure for Experimentation,” In Proceedings of the International Conference on Autonomous Agents, Workshop on Infrastructure for Scalable Multi-agent Systems, pp. 17–22, June 2–6, 2000, Barcelona, Spain.

148. Barber, K. S. and S. R. Jernigan. “Hybrid Domain Representation Archive (HyDRA) for Requirements Model Synthesis across Viewpoints,” In Proceedings of the International Conference on Software Engineering, p. 780, June 4–5, 2000, Limerick, Ireland.

149. Barber, K. S. and T. Graser. “Effective Representation and Search in Intelligent Requirements Management and Query Tools Supporting System Stakeholders,” (**Invited**) In Proceedings of the International Conference on Software Engineering, Third International Workshop on Intelligent Software Engineering, pp. 70–79, June 4, 2000, Limerick, Ireland.

150. Barber, K. S., R. McKay, A. Goel, D. Han, J. Kim, T. H. Liu, C. E. Martin. “Sensible Agents Capable of Adaptive Autonomy: The Distributed Architecture and Testbed,” (**Invited**) In Proceedings of the Virtual World and Simulation Conference, The Society for Computer Simulation International, pp 127–132, January 24–27, 2000, San Diego, CA.

151. Barber, K. S., A. Goel, and C. E. Martin. “The Motivation for Dynamic Adaptive Autonomy in Agent–Based Systems,” In Proceedings of the First Asia-Pacific Conference on Intelligent Agent Technology (IAT '99), **Best Paper Award**, pp. 131–140, December 14–17, 1999, Hong Kong.

152. Barber, K. S., T. J. Graser, P. Grisham, S. R. Jernigan, L. Mantock, and J. Silva. “The Knowledge-based Integrated Design and Development Environment (KIDDE): Integrating a Formal KA Process and Requirements Representation with a JAD/RAD Development Approach,” In Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling, and Management (KAW '99), pp. 1–7–1 through 1–7–18, October 16–21, 1999, Banff, Alberta, Canada.

153. Barber, K. S., R. M. McKay, C. E. Martin, T. H. Liu, J. Kim, D. Han, and A. Goel. “Sensible Agents in Supply Chain Management: An Example Highlighting Procurement and Production Decisions,” In Proceedings of the Internet-Aided Design, Manufacturing, and Commerce Technical Committee, 19th ASME Computers and Information in Engineering Conference, paper number CIE-9078 (Compact Disc), pp. 1-9, September 12–15, 1999, Las Vegas, NV.

154. Barber, K. S., T. J. Graser, and S. R. Jernigan. “The Systems Engineering Process Activities: Supporting Early Requirements Integration prior to Implementation Design,” In Proceedings of the Software Technology and Engineering Practice (STEP99), pp. 50–59. August 30–September 2, 1999, Pittsburgh, PA.

155. Barber, K. S., A. Goel, and C. E. Martin. “The Motivation for Dynamic Adaptive Autonomy in Agent - Based Systems,” In Proceedings of the 1999 International Joint Conference on Artificial Intelligence (IJCAI), Workshop on Adjustable Autonomy Systems, pp. 1–6, July 31–August 6, 1999, Stockholm, Sweden.

156. Barber, K. S. and S. R. Jernigan. “Changes in the Model Creation Process to Ensure Traceability and Reuse,” In Proceedings of the International Conference on AI – Software Engineering Track (IC-AI99), pp. 591–597, June 28–July 1, 1999, Las Vegas, NV.

157. Barber, K. S., A. Goel, D. Han, T. H. Liu, C. E. Martin, R. McKay, and J. Kim. “Problem-Solving Frameworks for Sensible Agents in an Electronic Market,” (**Invited**) In Proceedings of the Applications of Agent-based Systems Technology Workshop at the Twelfth International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE-99), pp. 470–479, May 31–Jun 3, 1999, Cairo, Egypt.

158. Barber, K. S., S. R. Jernigan, and T. J. Graser. “Integrating Third Party Artificial Intelligence Components into Mainstream Applications,” In Proceedings of the Twenty-First International Conference on Software Engineering (ICSE99) - Workshop Ensuring Successful COTS Development, pp. 1-6, May 16–22, 1999, Los Angeles, CA.

159. Barber, K. S., R. McKay, and T. H. Liu. “Group Membership Services for Dynamically Organized Sensible Agent-based Systems,” In Proceedings of the Twelfth International Florida Artificial Intelligence Research Society (FLAIRS'99), pp. 160–165, May 1–5, 1999, Orlando, FL.

160. Barber, K. S., T. H. Liu, A. Goel, and C. E. Martin. “Conflict Representation and Classification in a Domain-Independent Conflict Management Framework,” In Proceedings of the Third International Conference on Autonomous Agents, Agents’99, pp. 346–347, May 1–5, 1999, Seattle, WA.

161. Barber, K. S. and J. Kim. “Constructing and Dynamically Maintaining Perspective-based Agent Models in a Multi-agent Environment,” In Proceedings of the Third International Conference on Autonomous Agents, Agents’99, pp. 416–417, May 1–5, 1999, Seattle, WA.

162. Barber, K. S. and C. E. Martin. “Agent Autonomy: Specification, Measurement, and Dynamic Adjustment,” In Proceedings of the Autonomy Control Software Workshop, Agents '99, pp. 8–15, May 1–5, 1999, Seattle, WA.

163. Barber, K. S., T. J. Graser, S. R. Jernigan, and J. Silva, M.D. "Increasing Opportunities for Reuse through Tool and Methodology Support for Enterprise-Wide Requirements Reuse and Evolution," In Proceedings of the First International Conference on Enterprise Information Systems (ICEIS'99), Vol. 2, pp. 383–390, March 27–30, 1999, Setúbal, Portugal.

164. Barber, K. S., and C. E. Martin. "Applying Dynamic Planning Frameworks to Agent Goals," In Proceedings of the AAAI Spring Symposium Series 1999, Agents with Adjustable Autonomy, pp. 1–8, March 22–25, 1999, Stanford University, Stanford, CA.

165. Barber, K. S. "Dynamic Adaptive Autonomy in Agent-based Systems," (**Invited**) In Proceedings of the Fourth International Symposium on Autonomous Decentralized Systems (ISADS 99), pp. 402–405, March 21–23, 1999, Tokyo, Japan.

166. Barber, K. S., T. J. Graser, and S. R. Jernigan "Increasing Opportunities for Reuse through Tool and Methodology Support for Enterprise-Wide Requirements Reuse and Evolution," (**Invited**) In Proceedings of the 1999 Workshop on Institutionalizing Software Reuse (WISR 99), pp. 19–24, January 7–9, 1999, Austin, TX.

167. Barber, K. S., T. J. Graser, S. R. Jernigan, B. J. McGiverin, and J. Silva. "Application of the SEPA Methodology and Tool Suite to the National Cancer Institute," In Proceedings of the Thirty-Second Hawaiian International Conference on System Sciences, Special Track on Information Technology in Health Care, pp. 1-15 (Compact Disc), January 5–8, 1999, Maui, HI.

168. Barber, K. S., E. White, A. Goel, D. Han, J. Kim, H. Li, T. H. Liu, C. E. Martin, and R. McKay. " Dynamic, Self-Organizing, Multi-Agent based Shop Floor Control," (**Invited**) In Proceedings of the Fourth Joint International Conference on Information Sciences (IJCIS), Workshop on Intelligent Control, pp. 306–312, October 23–28, 1998, Research Triangle Park, NC.

169. Barber, K. S. and D. C. Han. "Multi-Agent Planning under Dynamic Adaptive Autonomy," In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 1, pp. 399–404, October 11–14, 1998, San Diego, CA.

170. Barber, K. S. and R. M. McKay. "Allocating Goals and Planning Responsibility in Dynamic Sensible Agent Organizations," In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 1, pp. 405–410, October 11–14, 1998, San Diego, CA.

171. Barber, K. S., T. H. Liu, A. Goel, and S. Ramaswamy. "A Flexible Reasoning Mechanism for the Trade-off of System Versus Local Goals in Sensible Agents," In Proceedings of the International Conference on Agile, Intelligent and Computer–Integrated Manufacturing. CD ROM, October 7–9, 1998, Troy, NY.

172. Barber, K. S., E. R. White, A. Goel, D. Han, J. Kim, T.H. Liu, C. E. Martin, and R. McKay. "Sensible Agent Problem Solving Simulation for Manufacturing Environments," (**Invited**) In Proceedings of the AAAI Artificial Intelligence and Manufacturing Research Planning Workshop: State of the Art and State of the Practice, pp. 1–8, August 31 – September 2, 1998, Albuquerque, NM.

173. Barber, K. S., T. J. Graser, S. R. Jernigan, and B. J. McGiverin. "Features of the Systems Engineering Process Activities (SEPA) Methodology," (**Invited**) In Proceedings of the AAAI Artificial Intelligence and Manufacturing Research Planning Workshop: State of the Art and State of the Practice, pp. 9–15, August 31 – September 2, 1998, Albuquerque, NM.

174. Barber, K. S., T.J. Graser, S.R. Jernigan, B.J. McGiverin, and E. R. White. "The Application of the Systems Engineering Process Activities in the Population of Conceptual Models of the Mission Space," In Proceedings of the Conference of the Society of Computer Simulation, pp. 111–117, July 19–22, 1998, Reno, NV.

175. Barber, K. S. and A. Goel. "The Classification and Specification of a Domain Independent Agent Architecture," (**Invited**), In Proceedings of the Eleventh International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE-98), Vol. 1, pp. 568–576, June 1–4, 1998, Benicassim, Castellon, Spain.

176. Martin, C. E. and K. S. Barber. "Flexible Organizational Roles for Autonomous Agents," In Proceedings of the Workshop on Agent-based Manufacturing at the Second International Conference Autonomous Agents, Agents'98, pp. 79–86, May 10, 1998, Minneapolis, MN.

177. Goel, A., K. S. Barber, T. H. Liu, and E. White. "Implementing Sensible Agents in a Distributed Simulation Environment," In Proceedings of the 1998 Conference on Virtual Worlds and Simulation, Society of Computer Simulation, pp. 206 – 211, January 11–14, 1998, San Diego, CA.

178. Barber, K. S. "Sensible Agents," (**Invited**) In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 5, pp. 4146–4151, October 12–15, 1997, Orlando, FL.

179. Barber, K. S. "Adaptive Autonomy: The Key to Dynamic, Responsive Formation of Sensible Agent Organizations," (**Invited**) In Proceedings of the International Conference on Intelligent Systems and Semiotics, pp. 263-268, September 22–25, 1997, Gaithersburg, MD.

180. Graser, T., B. J. McGiverin, and K. S. Barber. “A Supply Chain Configuration Tool,” In Proceedings of the ASME Design Engineering Technical Conferences, Internet Aided Design, Manufacturing, and Commerce, Paper Number: DETC97/CIE-4293, pp.1 – 9, September 14–17, 1997, Sacramento, CA.

181. Chuter, C., C. Chase, and K. S Barber. “Sensible Agents in Discrete Event Virtual Environment Simulation,” In Proceedings of the ISMCR ‘97: Topical Workshop on Virtual Reality and Advanced Man-Machine Interfaces, Vol. IXB, topic 17, pp. 79–85, June 1–6, 1997, Tampere, Finland.

182. Liu, T.H., C. Chuter, and K. S. Barber. “Virtual Environment Simulation for Visualizing Conflict Resolution Strategies in Multiple Robot Systems,” In Proceedings of the IASTED International Conference, Robotics and Manufacturing, pp. 154–158, May 29–31, 1997, Cancun, Mexico.

183. Ramaswamy, S., A. Suraj, and K. S. Barber. “An Approach for Monitoring and Control of Agent-based Systems,” In Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 4, pp. 3467–3472, April 20–25, 1997, Albuquerque, N.M.

184. Martin, C. E. and K.S. Barber. “Multiple, Simultaneous Autonomy Levels for Agent-based Systems,” In Proceedings of the Fourth International Conference on Control, Automation, Robotics and Vision, Vol. 2, pp. 1318–1322, December 3–6, 1996, Singapore.

185. Suraj, A., S. Ramaswamy, and K. S. Barber. “Extended Statecharts: A Specification Formalism for High Level Design,” In Proceedings of the Fourth International Conference on Control, Automation, Robotics and Vision, Vol. 1, pp. 53–57, December 3–6, 1996, Singapore.

186. Barber, K. S. “The Architecture for Sensible Agents,” In Proceedings of the Semiotic Modeling for Sensible Agents Workshop, Intelligent Systems Conference, Vol. 2, pp. 49–54, October 23–25, 1996, Gaithersburg, MD.

187. Martin, C. E. and K. S. Barber. “Representation of Autonomy in Distributed Agent-based Systems,” In Proceedings of the Semiotic Modeling for Sensible Agents Workshop, Intelligent Systems Conference, Vol. 2, pp. 67–72, October 23–25, 1996, Gaithersburg, MD.

188. Graser, T. and K. S. Barber. “Meta-Modeling of Sensible Agents,” In Proceedings of the Semiotic Modeling for Sensible Agents Workshop, Intelligent Systems Conference, Vol. 2, pp. 61–66, October 23–25, 1996, Gaithersburg, MD.

189. Suraj, A., S. Ramaswamy, and K. S. Barber. “Behavioral Specification of Sensible Agents,” In Proceedings of the Semiotic Modeling for Sensible Agents Workshop, Intelligent Systems Conference, Vol. 2, pp. 55–60, October 23–25, 1996, Gaithersburg, MD.

190. Goel, A., T.H. Liu, and K.S. Barber. “Conflict Resolution in Sensible Agents,” In Proceedings of the Semiotic Modeling for Sensible Agents Workshop, Intelligent Systems Conference, Vol. 2, pp. 80–85, October 23–25, 1996, Gaithersburg, MD.

191. Macfadzean, R. and K. S. Barber. “Reasoning about Autonomy in Multi-Agent Systems,” In Proceedings of the Semiotic Modeling for Sensible Agents Workshop, Intelligent Systems Conference, Vol. 2, pp. 73–79, October 23–25, 1996, Gaithersburg, MD.

192. Macfadzean, R. and K. S. Barber. “Radar Frequency Assignment in Mobile Radar Units,” In Proceedings of the Semiotic Modeling for Sensible Agents Workshop, Intelligent Systems Conference, Vol. 2, pp. 86–91, October 23–25, 1996, Gaithersburg, MD.

193. Chuter, C., S. R. Jernigan, and K. S. Barber. “A Virtual Environment Simulator for Reactive Manufacturing Schedules,” In Proceedings of the Symposium on Virtual Reality in Manufacturing Research and Education, pp. 197–209, October 7–8, 1996, Chicago, IL.

194. Martin, C. E., R. H. Macfadzean, and K. S. Barber. “Supporting Dynamic Adaptive Autonomy for Agent-based Systems,” In Proceedings of the Artificial Intelligence and Manufacturing Research Planning Workshop, (Ed. George F. Luger), AAAI Press, pp. 112–120. June 24–26, 1996, Albuquerque, NM.

195. Ramaswamy, S., K. P. Valavanis, and K. S. Barber. “Model Development of MIMO Nets: A H-EPN Based Approach,” In Proceedings of the 1995 IEEE International Conference on Decision and Control, Vol. 1, pp. 897–904, December 13–15, 1995, New Orleans, LA.

196. Suraj, A., S. Ramaswamy, and K. S. Barber. “An Integrated Approach to Design Reusability,” In Proceedings of the Conference on Integrated Design and Process Technology, Vol. 1, pp. 326–331, December 7–9, 1995, Austin, TX.

197. Barcio, B. T., S. Ramaswamy, and K. S. Barber. “An Object-Oriented, Model-Based Approach to Software Systems Development,” In Proceedings of the Fifth International Conference on Software Quality, pp. 30–41, October 23–26, 1995, Austin, TX.

198. Jernigan, S.R., S. Ramaswamy, and K. S. Barber. “On-Line Scheduling Using a Distributed Simulation Technique for Intelligent Manufacturing Systems,” In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 3, pp. 2159–2164, October 22–25, 1995, Vancouver, Canada.

199. Barcio, B.T., S. Ramaswamy, R. H. Macfadzean, and K. S. Barber. “Object-Oriented Analysis, Modeling and Simulation of a Notional Air Defense System,” In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 5, pp. 3983–3988, October 22–25, 1995, Vancouver, Canada.

200. Macfadzean, R.H. and K. S. Barber. “An Approach for Decision-making and Control in Geographically Distributed Systems,” In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 4, pp. 3816–3821, October 22–25, 1995, Vancouver, Canada.

201. Ramaswamy, S. and K. S. Barber. “A High Level Specification Mechanism for Analysis and Design of Manufacturing Systems,” In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 1, pp. 524–529, October 22–25, 1995, Vancouver, Canada.

202. Chuter, C., S. Ramaswamy and K. S. Barber. “A Virtual Environment for Construction and Analysis of Manufacturing Prototypes,” In Proceedings of the ASME International Computers in Engineering Conference, Special Session on Virtual Environments and Systems for Product Development, pp. 927–932, September 17–21, 1995, Boston, MA.

203. Barber, K. S. “A Feature-based CAD Representation Enabling Case-based Planning Across Manufacturing Applications,” In Proceedings of the 1995 IEEE International Symposium on Intelligent Control, pp. 345–350, August 27–29, 1995, Monterey, CA.

204. Macfadzean, R. H., K. S. Barber, and S. A. Szygenda. “Discrete Event Simulation of Radar Interference,” In Proceedings of the 1995 Summer Computer Simulation Conference, pp. 489–494, July 24–26, 1995, Ottawa, Ontario, Canada.

205. Macfadzean, R. H. and K. S. Barber. “Simulation of Multi-Layer Belief Nets for Situation Assessment,” In Proceedings of the 1995 Summer Computer Simulation Conference, pp. 483–488, July 24–26, 1995, Ottawa, Ontario, Canada.

206. Barcio, B. T., S. Ramaswamy, and K. S Barber. “OARS: An Object Oriented Architecture for Reactive Systems,” In Proceedings of the 1995 IEEE International Conference on Robotics and Automation, Vol. 1, pp. 1093–1098, May 21–27, 1995, Nagoya, Japan.

207. Swaminathan, A., and K. S. Barber. “APE: An Experience-based Assembly Sequence Planner for Mechanical Assemblies,” In Proceedings of the 1995 IEEE International Conference on Robotics and Automation, Vol. 2, pp. 1278–1283, May 21–27, 1995, Nagoya, Japan.

208. Shaikh, S., J. Bradley, S. A. Szygenda, and K. S. Barber. “Modeling Errors at a Hierarchical Level for Design Verification,” In Proceedings of the IASTED International Conference on Modeling and Simulation, pp. 379–381, April 27–29, 1995, Pittsburgh, PA.

209. Barber, K. S. “A Feature-based CAD Representation Enabling Case-based Planning across Multiple Manufacturing Applications,” In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 1, pp. 142–147, October 2–5, 1994, San Antonio, TX.

210. Barber, K. S., O. R. Mitchell, and K. A. Harbison-Briggs. “Symbolic Feature-Based Representation and Planning for an Agent Based Robot Controller,” In Proceedings of the IEEE International Symposium on Intelligent Controls, pp. 475–480, August 13, 1991, Arlington, VA.

211. Barber, K. S., J. C. M. Tiernan, and K. A. Harbison-Briggs. “A Process Model for an Intelligent Robotic Planning System,” In Proceedings of the Modeling and Simulation Conference, Vol. 22, No. 2, pp. 1013–1020, May 4–5, 1991, Pittsburgh, PA.

212. Barber, K. S., L. Godwin, O. R. Mitchell, and R. Button. “CAD-Based Planning for Robot Control,” In Proceedings of the Symposium on the Control of Robots and Manufacturing Systems, pp. 150–158, November 9, 1990, Fort Worth, TX.

213. Mullen, D., R. Duff, and K. S. Barber. “Part Design Data Driven Trajectory Planner using Parallel Processing,” In Proceedings of the SME Aerospace Automation Conference, pp. MS90-275 - MS90-275-10, October 1990, Arlington, TX.

214. Tiernan, J. C. M., K. S Barber, and K. A. Harbison-Briggs. “Knowledge Acquisition of Process Knowledge for Intelligent Robotic Manufacturing Applications,” In Proceedings of the Midcon 1990, pp. 240–243, September 11–13, 1990, Dallas, TX.

215. Tiernan, J. C. M., K. S Barber, and K. A. Harbison-Briggs. “Integration of Blackboard-Based Process Knowledge with Conventional Robot Control Functions on Parallel Hardware,” In Proceedings of the AAAI Blackboard Systems Workshop, July 29 – August 3, 1990, Boston, MA.

216. Tiernan, J. C. M., K. S Barber, and K. A. Harbison-Briggs. “Using Process Knowledge for Intelligent Robotic Manufacturing Applications,” In Proceedings of the AAAI Manufacturing Planning and Control Workshop, July 29 – August 3, 1990, Boston, MA.

217. Barber, K. S. and K. A. Harbison-Briggs. “The Structure of a Knowledge Base for Flexible Assembly,” In Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Integrated Architectures for Manufacturing, pp. 51–53, August 24, 1989, Detroit, MI.

218. Barber, K. S., D. Mullen, and O. R. Mitchell. “Design to Manufacturing Interface -- Intelligent Kinematic Engine,” In Proceedings of the Robotics and Expert Systems Symposium (ROBEXS), Vol. 4, pp. 293–298, August 2–4, 1989, Palo Alto, CA.

219. Barber, K. S. and D. Mullen. “Intelligent Kinematic Engine (IKE) for a Robot Controller,” In Proceedings of the Modeling and Simulation Conference, Vol. 20, No. 5, pp. 1805–1809, May 4–5, 1989, Pittsburgh, PA.

220. Barber, K. S. “Tutorial on Compliant Control of Robotic Manipulators for Assembly Tasks,” In Proceedings of the Midcon 1988 , pp. 363–366, August 1988, Dallas, TX.

221. Barber, K. S. “Interactive Machine Learning Language (IML2) – Applications for Assembly Tasks,” In Proceedings of the Modeling and Simulation Conference, Vol. 19, No. 2, pp. 583–589, May 4–5, 1988, Pittsburgh, PA.

222. Barber, K. S. and G. J. Agin. “Analysis of Human Communication During Assembly Tasks,” In Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 3, pp. 1524–1529, March 31–April 3, 1987, Raleigh, NC.

<u>Other Conference Proceedings</u>

1. K. Suzanne Barber, “The Identity of the Insider Threat,” Defense Innovation Summit, Austin Texas, November 29 - December 1, 2016.
2. Zaeem, R. Nokhbeh and K. Suzanne Barber, “Trust Filtering to find the Best Sources in Social Media,” Defense Innovation Summit, Austin, Texas, November 29 - December 1, 2016.
3. Zaiss, James and K. Suzanne Barber, “Identity Threat Assessment and Prediction,” Defense Innovation Summit, Austin, Texas, November 29 - December 1, 2016.
4. Brenner, J. and K.S. Barber, “Identity Threat Assessment and Prediction,” ID360: The Global Forum on Identity, April 30-May 1, 2013, Austin, Texas.
5. Budalakoti, S. K.S. Barber, M. Rasheed, and C. Bajaj, “A Bayesian Network based Framework for Identity Risk Management,” ID360: The Global Forum on Identity, April 30-May 1, 2013, Austin, Texas.
6. Soeder, B. and K.S. Barber, “Towards a Metric for Identity Confidence using an Agent Approach,” ID360: The Global Forum on Identity, April 30-May 1, 2013, Austin, Texas.
7. Yang, Y. and K.S. Barber, “Facebook Privacy Checker,” ID360: The Global Forum on Identity, April 30-May 1, 2013, Austin, Texas.
8. "The Financial Impact of Breached Protected Health Information: A Business Case for Enhanced PHI Security" ANSI. 2012. (Contributor: K. Suzanne Barber)
9. Fullam, K., T. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. Rosenschein, L. Vercouter, and M. Voss. “A Demonstration of The Agent Reputation and Trust (ART) Testbed: Experimentation and Competition for Trust in Agent Societies,” In Proceedings of International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005) Demonstration Track, pp. 151-152, July 25-29, 2005, Utrecht, Netherlands.
10. Barber, K.S. “Trustworthiness Assessment, Information Valuation and Coordinated Action Selection for Multi-Agents Controlling UAV Surveillance,” Special Seminar on Defense Applications of Agent Systems (DAAS) at The Third International Joint Conference on Autonomous Agents and Multi Agent Systems, July 21, 2004, New York City, NY.
11. Barber, K.S., “Robust and Stable Decision-Making despite Operations in Chaotic Information and Kinetic Warfare Environments,” DARPA Augmented Cognition Program meeting, January 6, 2004, Orlando, FL.
12. Barber, K.S., K. Fullam, T. Graser, D. Han, J. Kim, D. Lam, M. MacMahon, and R. McKay. “Sensible Agent Technology Improving Coordination and Communication in Biosurveillance Domains,” 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS2002), Demonstration Program, July 15-19, 2002, Bologna, Italy.
13. Barber, K. S., I.M. Gamba, A. Goel, D. C. Han, J. Kim, D. N. Lam, M. MacMahon, C. E. Martin, and R. McKay. “Sensible Agent Capabilities,” Proceedings on CD-ROM, DARPA TASK Principal Investigator meeting, April 17-19, 2001, Albuquerque, NM.

14. Arapostathis, A., K. S. Barber, L.A. Caffarelli, I.M. Gamba, A. Goel, D. C. Han, J. Kim, D. N. Lam, M. MacMahon, C. E. Martin, and R. McKay. “Planned Application of Sensible Agents to the Virtual Transportation Company Domain,” Proceedings on CD-ROM, DARPA TASK Principal Investigator meeting, April 17-19, 2001, Albuquerque, NM.
15. Barber, K. S., M. MacMahon, R. McKay, A. Goel, D. C. Han, J. Kim, D. N. Lam, and C. E. Martin. “An Agent Infrastructure Implementation for Leveraging and Collaboration in Operational and Experimental Environments,” Proceedings on CD-ROM, DARPA TASK Principal Investigator meeting, April 17-19, 2001, Albuquerque, NM.
16. Barber, K. S. and J. Kim. “Constructing and Dynamically Maintaining Perspective-based Agent Models for Command and Control Applications in a Multi-Agent Environment,” In Proceedings of the DARPA-JFACC Symposium on Advances in Enterprise Control, DARPA-ISO, pp. 161–170, November 15–16, 1999, San Diego, CA.
17. Barber, K. S. “Model for Constructing Manufacturing Virtual Environments,” In Proceedings of the 1996 NSF Design and Manufacturing Grantees Conference, pp. 111–112, January 2–5, 1996, Albuquerque, NM.
18. Barber, K. S. “The Stability Analysis of Autonomous Agents in Dynamic, Complex Manufacturing Environments,” In Proceedings of the 1996 NSF Design and Manufacturing Grantees Conference, pp. 181–182, January 2–5, 1996, Albuquerque, NM.
19. “The Automated Factory of the Future: Where Do We Go From Here?,” In Proceedings of the 1994 IEEE International Conference on Robotics and Automation (Eds. Dr. Suzanne Barber and Dr. Erik Mettala), May 9, 1994, San Diego, CA.

Recent Technical Reports

1. “The Future Needs Identity Professionals,” K.S. Barber. UT CID Report#: 20-16, July 2020.
2. “How Much Identity Management with Blockchain Would Have Saved Us? R.N. Zaeem and K. S. Barber. UT CID Report#: 20-15, July 2020.
3. “Negative Sentiment Is a [Statistically] Significant Indicator of Fake News,” R.N. Zaeem, C. Li, and K. S. Barber. UT CID Report#: 20-14, July 2020.
4. “On Sentiment of Online Fake News,” R.N. Zaeem, C. Li, and K. S. Barber. UT CID Report#: 20-13, July 2020.
5. “A Survival Game Analysis to Common Personal Identity Protection Strategies,” D. Liau, R.N. Zaeem, and K.S. Barber. UT CID Report#: 20-12, June 2020.
6. “A Framework for Estimating Privacy Risk Scores of Mobile Apps,” K. C. Chang, R.N. Zaeem, and K.S. Barber. UT CID Report#: 20-11, June 2020.
7. “PrivacyCheck’s Machine Learning to Digest Privacy Policies: Competitor Analysis and Usage Patterns”, R.N. Zaeem, Anya S., Issa, A., Nimergood, J. Rogers, I., Shah, V., Srivastava, A., and K.S. Barber. UT CID Report#: 20-10, June 2020
8. “Election Prediction with Trust Filters,” Huang, T., R.N. Zaeem, and K.S. Barber. UT CID Report#: 20-09, June 2020.
9. “Tracking COVID-19 with Social Media,” Zaeem, R.N., C. Li, and K. S. Barber. UT CID Report#: 20-08, June 2020.
10. “PrivacyCheck v2: A Tool that Recaps Privacy Policies for You” R.N. Zaeem, Anya S., Issa, A., Nimergood, J. Rogers, I., Shah, V., Srivastava, A., and K.S. Barber. UT CID Report#: 20-07, June 2020
11. “Privacy, a Machine Learning Perspective,” R.N. Zaeem and K. S. Barber. UT CID Report#: 20-06, May 2020.
12. “Digital Identity Theft and Fraud,” R.N. Zaeem and K. S. Barber. UT CID Report#: 20-05, May 2020.
13. “Comparing Privacy Policies of Government Agencies and Companies,” R.N. Zaeem and K. S. Barber. UT CID Report#: 20-04, March 2020.
14. “The Effect of the GDPR on Privacy Policies: Recent Progress and Future Promise,” R.N. Zaeem and K. S. Barber. UT CID Report#: 20-03, March 2020.
15. “Is Your Phone You?” K.C. Chang, R.N. Zaeem, and K.S. Barber. UT CID Report#: 20-02, March 2019.
16. “Identifying Real-World Credible Experts in the Financial Domain to Avoid Fake News,” T. Huang, R.N. Zaeem, and K.S. Barber., UT CID Report#: 20-01, March 2020.
17. “Privacy in the Digital Age,” K.S. Barber. UT CID Report#: 19-09, October 2019.
18. “The Identity Ecosystem,” R.N. Zaeem, D. Liau, S. Budalakoti, and K.S. Barber. UT CID Report#: 19-08, July 2019
19. 2019 ITAP Report, J. Zeiss, R. Anderson. R. Nokhbeh Zaeem, K. Suzanne Barber, UT CID Report #19-07, July, 2019.

20. "An Assessment of Blockchain Identity Solutions: Minimizing Risk and Liability of Authentication," R. Rana, R.N. Zaeem, and K.S. Barber. UT CID Report#: 19-06, July 2019.
21. "An Evaluation Framework for Future Privacy Protection Systems: A Dynamic Identity Ecosystem Approach," D. Liau, R.N. Zaeem, and K.S. Barber. UT CID Report#: 19-05, July 2019.
22. "Statistical Analysis of Identity Risk of Exposure and Cost Using the Ecosystem of Identity Attributes," C. Chen, R. N. Zaeem, and K.S. Barber. UT CID Report#: 19-04, April 2019.
23. "Enhancing and Evaluating Identity Privacy and Authentication Strength by Utilizing the Identity Ecosystem," K.C. Chang , R.N. Zaeem, K. S. Barber. UT CID Report#: 19-03, April 2019.
24. "Internet of Things: Securing the Identity by Analyzing Ecosystem Models of Devices and Organizations," K.C. Chang , R.N. Zaeem, K. S. Barber. UT CID Report#: 19-02, April, 2019.
25. "US-Centric vs. International Personally Identifiable Information: A Comparison Using the UT CID Identity Ecosystem," R. Rana, R.N. Zaeem, and K.S. Barber. UT CID Report#: 19-01, April 2019.
26. "Predicting Disease Outbreaks Using Social Media: Finding Trustworthy Users," D. Liau, R.N. Zaeem, K. S. Barber. UT CID Report#: 18-07 November 2018.
27. "Identity Threat Assessment and Prediction," J. Zaiss, R.N. Zaeem, K. S. Barber. UT CID Report#: 18-06 September 2018.
28. "Understanding Victim-enabled Identity Theft: Perpetrator and Victim Perspectives," Lacey, D., K. S. Barber., and James Zaiss, May 4, 2018.
29. "Current Biometric Adoption and Trends," German, R., and K.S. Barber. May 1, 2018.
30. "Consumer Attitudes about Biometric Authentication," German, R., and K.S. Barber. May 1, 2018
31. "2017 Identity Threat Assessment and Prediction (ITAP) Report," April 20, 2017
32. "A study of Web Privacy Policies Across Industries," Zaeem, R. Nokhbeh and K.S. Barber. April 13, 2017.

Other Publications – Invited Articles and Media Coverage of the Center for Identity

1. "Find out when your online accounts are compromised -- for free**"** https://cbsaustin.com/news/local/find-out-when-your-online-accounts-are-compromised-for-free. CBS News Austin, January 31, 2020.
2. "Facial Recognition's Arrival at DFW Airport Ushers in the Biometric Era," https://www.dallasnews.com/business/airlines/2019/09/25/facial-recognition-s-arrival-at-dfw-airport-ushers-in-the-biometric-era/, September 25, 1 2019
3. "2019 Identity Theft Report Released," https://www.itij.com/latest/news/2019-identity-theft-report-released, International Travel and Health Insurance Journal, July 31, 2019
4. "Biometrics: Dismantling Myths Surrounding Facial Recognition," https://www.securityweek.com/biometrics-dismantling-myths-surrounding-facial-recognition, SecurityWeek.com, Jul 18, 2019
5. "Digital Dilemma: Is Austin a Cybersecurity Hub? Depends on Who You Ask," https://www.statesman.com/news/20190228/digital-dilemma-is-austin-cybersecurity-hub-depends-who-you-ask, Austin-American Statesman, February 28 2019
6. "Gartner: Future of identity management is mobile, SaaS," https://securitboulevard.com/2019/02/gartner-future-of-identity-management-is-mobile-saas/, Security Boulevard, February 21, 2019
7. "New On Parent's To-Do List: Checking Children's Credit History," https://www.wsj.com/articles/new-on-parents-to-do-list-checking-childrens-credit-history-1535457603, Wall Street Journal, January 17 2019
8. "How AI Can Help Stop Cyberattacks," https://www.wsj.com/articles/how-ai-can-help-stop-cyberattacks-1537322940, Wall Street Journal, September 18, 2018
9. "Public embraces biometrics but with reservations," https://www.meritalk.com/articles/study-public-embraces-biometrics-with-reservations/ Meritalk.com, May 21, 2018
10. "The 'old-fashioned' identity theft is just as dangerous as the cyber kind," https://www.cnbc.com/2018/05/11/non-digital-identity-theft-can-be-as-damaging-as-breaches-from-hacking.html , CNBC.com, May 15, 2018.
11. "Report shows consumer comfort with biometrics increasing but uneven," https://www.biometricupdate.com/201805/report-shows-consumer-comfort-with-biometrics-increasing-but-uneven Biometric Update.com, May 21, 2018

12. "New Survey on Biometric Technology Shows Consumers are OK with Some Forms and Wary of Others, https://news.utexas.edu/2018/05/03/new-survey-on-consumer-attitudes-toward-biometric-technology UT News, May 4, 2018
13. "What data are private companies collecting about you," http://cbsaustin.com/news/local/what-data-are-private-companies-collecting-about-you CBS Austin, February 2, 2018.
14. "Nondigital, Analog Theft is Main Driver in Identity-Related Crimes," https://news.utexas.edu/2017/04/24/nondigital-analog-theft-is-main-driver-in-identity-theft UT News, December 4, 2017
15. "Science delivers input for identity domain," Keesing Journal of Documents and Identity, November 1, 2017
16. "UT Center for Identity releases report on identity theft," https://www.dailytexanonline.com/2017/05/01/ut-center-for-identity-releases-report-on-identity-theft, May 2, 2017.
17. "6 Factors Impacting Identity Theft Risks, Credit Union Times, http://www.cutimes.com/2017/04/21/6-factors-impacting-identity-theft-risks, April 21, 2017.
18. "6 factors impacting identity theft risks," Property Casuality 360, http://www.propertycasualty360.com/2017/04/20/6-factors-impacting-identity-theft-risks?eNL=58f7ca23160ba0fd177366e7&utm_source=PC360_NewsFlash&utm_medium=EMC-Email_editorial&utm_campaign=04202017&page_all=1, April 20, 2017.
19. "8 Tips to Protect Your Identity Beyond the Computer, UTNews, https://news.utexas.edu/2017/04/28/8-tips-to-protect-your-identity-beyond-the-computer, April 28, 2017.
20. "Privacy Made Simple," https://privacymadesimple.net/tag/university-of-texas-at-austin-center-for-identity/, May 5, 2017.
21. "How Social Security Numbers became Skeleton Keys for Fraudsters," Christian Science Monitor, http://www.csmonitor.com/World/Passcode/Security-culture/2016/1121/How-Social-Security-numbers-became-skeleton-keys-for-fraudsters, November 21, 2016.
22. "How the Secret Service is fighting identity theft,"Practically Unhackable, https://medium.com/un-hackable/beyond-passwords-a-glimpse-into-the-world-of-offline-identity-theft-5340c3b046a8, May 5, 2016.
23. "Study: People with low health literacy don't find health apps helpful," http://www.dailytexanonline.com/2016/10/25/study-people-with-low-health-literacy-don%E2%80%99t-find-health-apps-helpful?platform=hootsuite, October 25, 2016.
24. "Tips to keep your Uber account secure," http://keyetv.com/news/local/tips-to-keep-your-uber-account-secure, January 20, 2016.
25. "6 Factors Impacting Identity Theft Risks, Credit Union Times, http://www.cutimes.com/2017/04/21/6-factors-impacting-identity-theft-risks, April 21, 2017.
26. "6 factors impacting identity theft risks," Property Casuality 360, http://www.propertycasualty360.com/2017/04/20/6-factors-impacting-identity-theft-risks?eNL=58f7ca23160ba0fd177366e7&utm_source=PC360_NewsFlash&utm_medium=EMC-Email_editorial&utm_campaign=04202017&page_all=1, April 20, 2017.
27. "8 Tips to Protect Your Identity Beyond the Computer, UTNews, https://news.utexas.edu/2017/04/28/8-tips-to-protect-your-identity-beyond-the-computer, April 28, 2017.
28. "UT Center for Identity releases report on identity theft," https://www.dailytexanonline.com/2017/05/01/ut-center-for-identity-releases-report-on-identity-theft, May 2, 2017.
29. "Privacy Made Simple," https://privacymadesimple.net/tag/university-of-texas-at-austin-center-for-identity/, May 5, 2017.
30. "DNC hack part of a cyber war that's just begun," San Antonio Express, http://www.expressnews.com/news/local/article/DNC-hack-part-of-a-cyber-war-that-s-just-begun-8748798.php, July 30, 2016.
31. K. Suzanne Barber, "The link between identity theft and terrorism," Austin American Statesmen, http://www.mystatesman.com/news/news/opinion/barber-the-link-between-identity-theft-and-terrori/npZNm/, December 1, 2015.
32. "Center for Identity releases new browser extension to simplify terms of agreement contracts," Individual.com, http://www.individual.com/storyrss.php?story=204678874&hash=7a244c17474a397d04ddb4b094a7ee03 , May 6, 2015.

33. "University of Texas: Identity theft happens every 2 seconds," KXAN, http://kxan.com/2015/05/05/identity-theft-occurs-every-2-seconds-according-to-ut/, May 5, 2015.
34. K. Suzanne Barber, "The Age of the Data Breach: A Plan for CEOs," Texas CEO Monthly,http://texasceomagazine.com/departments/age-data-breach/, January 17, 2015. "UT launches new identity security masters program," Austin Business Journal, http://www.bizjournals.com/austin/blog/techflash/2015/05/ut-launches-new-identity-security-masters-program.html, May 15, 2015.
35. "Free Tax Help, Abbot Hopes to Reduces Taxes," KEYE, http://www.keyetv.com/news/features/top-stories/stories/free-tax-help-abbott-hopes-reduce-taxes-businesses-25333.shtml, April 15, 2015.
36. "6 steps to protect a very small business from ID theft," Credit Cards.com, http://www.creditcards.com/credit-card-news/6-steps-protect-small-business-id-theft-1269.php, March 9, 2015.
37. "Here's how to disable location tracking on your phone," Quartz, 01/20/15, qz.com/309226/heres-how-to-disable-location-tracking-on-your-phone/?=1, January 20, 2015.
38. "The Age of the Data Breach: A Plan for CEOs," Texas CEO Monthly, 01/17/15, http://texasceomagazine.com/departments/age-data-breach/, January 17, 2015.
39. K. Suzanne Barber, "Proving Your Identity At The Doctor's Office: An Imperfect System," CSID Blog, http://www.csid.com/2014/10/proving-identity-doctors-office-imperfect-system/, October 29, 2014.
40. "Credit card companies making strides in security," KXAN, http://kxan.com/2014/10/26/credit-card-companies-making-strides-in-security, October 26, 2014.
41. "ID Wise from the Center for Identity," Federation of Genealogical Societies, http://www.fgs.org/rpac/2014/10/23/id-wise-from-the-center-for-identity/, October 23, 2014.
42. "Comptroller Combs Lauds University of Texas Identity Protection Efforts," Texas Insider, 10/09/14, http://www.texasinsider.org/comptroller-susan-combs-lauds-the-university-of-texas-at-austins-identity-protection-efforts/, October 9, 2014.
43. K. Suzanne Barber, "Identity Management: Gain control over your personal data with IDWise," LexisNexis Fraud of the Day, http://www.fraudoftheday.com/2014/10/07/guest-writer-suzanne-barber/, October 7, 2014.
44. "UT launches identity theft resource center," The Daily Texan, http://www.dailytexanonline.com/2014/10/08/ut-launches-identity-theft-resource-center, October 8, 2014.
45. "Barber: We must develop better ways to stop cybersecurity threats," Austin-American Statesman, http://www.mystatesman.com/news/news/opinion/barber-we-must-develop-better-ways-to-stop-cyberse/nhdRw/?icmp=statesman_internallink_textlink_apr2013_statesmanstubtomystatesman_launch#a357b9a8.3859003.735514, October 7, 2014.
46. K. Suzanne Barber, "What You Should Know About Those Apps Your Kids Want to Download," Austin American-Statesman, October 2, 2014.
47. "They Grow Up So Fast! Why Kids Are Learning To Think Like Tech CEOs Under Cyber Attack," Fast Company, http://www.fastcompany.com/3036259/elasticity/they-grow-up-so-fast-why-kids-are-learning-to-think-like-tech-ceos-under-cyber-at, October 2, 2014.
48. "Credit monitoring becomes a standard offer after breaches," Marketplace, http://www.marketplace.org/topics/your-money/credit-monitoring-becomes-standard-offer-after-breaches, September 9, 2014.
49. UT Creates New Tool to Protect Against Identity Theft, KEYE-TV, http://www.keyetv.com/news/features/top-stories/stories/ut-creates-new-tool-protect-against-identity-theft-18529.shtml, June 4, 2014.
50. "Child Identity Theft on the Rise," KEYE, http://www.keyetv.com/news/features/top-stories/stories/child-identity-theft-rise-18049.shtml#.U3N9vF7Z70A, May 9, 2014.
51. "How to Protect Your Social Security Number," Tom's Guide, http://www.tomsguide.com/us/how-to-protect-social-security-number,news-18741.html, May 7, 2014.
52. "Identity Thieves targeting kids," KSAT ABC 12, http://www.ksat.com/news/defenders/identity-thieves-targeting-kids/25827276, April 23, 2014.
53. "Universities beef up cybersecurity, identity theft research," GCN, http://gcn.com/blogs/pulse/2014/04/ut-uconn-cybersecurity-research.aspx, April 15, 2014.
54. K. Suzanne Barber, "COMMENTARY: How to ward off identity theft," The Monitor, http://www.themonitor.com/opinion/commentary-how-to-ward-off-identity-theft/article_6233afce-c424-11e3-99bb-001a4bcf6878.html, April 15, 2014

55. Michael Theis, "UT launches anti-ID theft, fraud initiative," Austin Business Journal, http://www.bizjournals.com/austin/news/2014/04/14/ut-launches-anti-id-theft-fraud-initiative.html, April 15, 2014.
56. UT identity theft resource center to open in summer 2014," The Daily Texan, http://www.dailytexanonline.com/news/2014/04/15/ut-identity-theft-resource-center-to-open-in-summer-2014 , April 15, 2015.
57. K. Suzanne Barber, "How to Ward Off Identity Theft," The Monitor, April 15, 2014.
58. K. Suzanne Barber, "Looking for a Tax Refund? So Are Identity Thieves," The Alcalde, http://alcalde.texasexes.org/2014/04/looking-for-a-tax-refund-so-are-identity-thieves/ , April 14, 2014.
59. Larry Magid, "Nearly A Fifth Of Americans Suffer Data Breach -- Many Risk ID Theft," Forbes, April 14, 2014.
60. K. Suzanne Barber, "Beware, tax time is ideal time for identity theft," Austin-American Statesman, http://www.statesman.com/news/news/opinion/barber-beware-tax-time-is-ideal-time-for-identity-/nfZQB/, April 14, 2014.
61. Associated Press, "University of Texas building ID theft web site" (Appeared: Dallas Morning News, ABC 7- http://abclocal.go.com/ktrk/story?section=news/state&id=9500171, Statesman - http://www.statesman.com/ap/ap/texas/university-of-texas-building-id-theft-web-site/nfXwm/, MyFoxAustin, CBS Local - http://dfw.cbslocal.com/2014/04/11/ut-building-id-theft-website/), April 11, 2014.
62. Larry Magid, "Identity theft is a problem from cradle to grave," Contra Costa Times, http://www.contracostatimes.com/news/ci_25540509/magid-identity-theft-is-problem-from-cradle-grave, April 11, 2014.
63. K. Suzanne Barber, "Taxpayers, watch out for identity theft," San Antonio Express News, http://www.mysanantonio.com/opinion/commentary/article/Taxpayers-watch-out-for-identity-theft-5393145.php, April 10, 2014.
64. Omar L. Gallaga, "UT Center for Identity announces details for $5 million ID theft resource center," Austin 360 Blog, http://www.austin360.com/weblogs/digital-savant/2014/apr/10/ut-center-identity-announces-details-5-million-id-/, April 10, 2014.
65. Omar L. Gallaga, "UT Center for Identity to fight ID theft with $5 million resource center," Austin-American Statesman, http://www.mystatesman.com/news/technology/ut-center-for-identity-to-fight-id-theft-with-5-mi/nfXj6/?icmp=statesman_internallink_invitationbox_apr2013_statesmanstubtomystatesmanpremium , April 10, 2014.
66. "UT holds 2-day forum on personal identity information protection ," KLBJ News, http://www.newsradioklbj.com/News/story.aspx?ID=2153039, April 9, 2014.
67. "The Financial Impact of Breached Protected Health Information: A Business Case for Enhanced PHI Security" American National Standards Institute (ANSI). 2012. (Contributor: K. Suzanne Barber)
68. K. Suzanne Barber, "Your Connected Identity Assets," Texas Perspectives, March 2014.

RESEARCH GRANTS AND CONTRACTS:

- "Identity in 5G," Experis with Verizon, 1/16/20 – 1/15/21, $520,761
- "UT Privacy-preserving Contact Tracing App" ECE Research Seed Grant, 6/1/20 – 8/31/20, $8,125.
- "UT CID Identity Ecosystem," Center for Identity Strategic Partners, Industrial Affiliate Program, PI: K. Suzanne Barber, 9/1/19 – 8/31/20, $150,000.
- "Identity Threat Assessment and Prediction Project," Center for Identity Strategic Partners, Industrial Affiliate Program, PI: K. Suzanne Barber, 9/1/19 – 8/31/20, $100,000.
- "Privacy Check," Center for Identity Strategic Partners, Industrial Affiliate Program, PI: K. Suzanne Barber, 9/1/19 – 8/31/20, $110,000.
- "Trust Filter," Center for Identity Strategic Partners, Industrial Affiliate Program, PI: K. Suzanne Barber, 9/1/19 – 8/31/20, $176,000.
- "Identity Leadership," Identity Theft Resource Center, 3/1/19 – 3/31/19, $30,000.
- "UT CID Identity Ecosystem," Center for Identity Strategic Partners, Industrial Affiliate Program, PI: K. Suzanne Barber, 9/1/18 – 5/31/19, $100,000.
- "Identity Threat Assessment and Prediction Project," Center for Identity Strategic Partners, Industrial Affiliate Program, PI: K. Suzanne Barber, 9/1/18 – 5/31/19, $100,000.

- "Privacy Check," Center for Identity Strategic Partners, Industrial Affiliate Program, PI: K. Suzanne Barber, 9/1/18 – 8/31/19, $60,000.
- "Trust Filter," Center for Identity Strategic Partners, Industrial Affiliate Program, PI: K. Suzanne Barber, 9/1/17 – 8/31/18, $50,000.
- Digital Futures, "Personas and Threats in the Identity Ecosystem – Feedback and Refinement," PI: K. Suzanne Barber, 10/1/17 – 6/31/18, $143,815.
- Department of Homeland Security (DHS) Office of Biometric Identity Management (OBIM), "Biometrics Opportunity Architecture: Biometrics Improving Quality of Life," PI: K. Suzanne Barber, 9/25/2017 – 12/26/2018, $573,390.
- TransUnion, "Consumer Biometrics," 12/1/15 – 5/31/17, $75,000. Digital Futures, "Personas and Threats in the Identity Ecosystem – Feedback and Refinement," PI: K. Suzanne Barber, 7/1/17 – 1/31/18, $143,815.
- Defense Threat Reduction Agency (DTRA), "Surety BioEvent App," PI: K. Suzanne Barber, Co-PIs: Lauren Meyers and Andy Ellington, 5/1/14 - 4/30/17, $2,771,939. (KSB Portion: $1,441,408)
- Island Financial, "Identity Values and Liabilities," 11/1/2015 – 12/31/15, $20,000.
- TransUnion, "Consumer Biometrics," 12/1/15 – 5/31/17, $75,000.
- MorphoTrust and National Institute of Standards and Technology (NIST), "Level of Assurance Evaluation," PI: K. Suzanne Barber, 10/1/14 – 2/28/15, $66,351
- State of Texas, "Trust Online: Identity Security & Privacy for Citizens & Businesses," PI: K. Suzanne Barber, 9/1/13 – 8/31/15, $5M.
- National Science Foundation (NSF), "IGERT: Sustainable Grid Integration of Distributed and Renewable Resources, National Science Foundation," Co-PIs: Ross Baldick, Tom Edgar, Alexis Kwasinski, Michael Weber, 7/1/10 – 6/30/15, $1,227,203.
- National Science Foundation (NSF), "Sustainable Grid Integration of Distributed and Renewable Resources," Co-PIs: Ross Baldick, Tom Edgar, Alexis Kwasinski, Michael Weber, 7/01/10 - 06/30/15, $630,806.
- "Systems Engineering for Test Evaluation Assessments," U.S. Army Operational Test Command, $100,000, 6/1/13 – 5/31/14.
- "IGERT: Sustainable Grid Integration of Distributed and Renewable Resources," National Science Foundation, PIs: Ross Baldick, K. Suzanne Barber, Tom Edgar, Alexis Kwasinski, Michael Weber, $3,129,768, 7/1/10 – 6/30/15.
- IBM, "Identity Analytics and Visualization," 5/1/12 – 8/31/12, $48,000.
- Department of Homeland Security (DHS), "Sensor and Social Networks Armed to Detect and Defend against Terrorist Attacks," PI: K. Suzanne Barber, 9/1/09 – 11/30/09, $40,000.
- Naval Undersea Warfare Center, "Wargaming and Investigations for Identifying Distributed Networking Technologies for Unmanned Distributed Networked Systems," PI: K. Suzanne Barber, 10/1/09 – 9/30/09, $80,000.
- DARPA, Building Problem Solving Teams for Dynamic Human and Agent Collaboration, 9/1/08 – 10/31/10, $189,000.
- Cougaar Software, "Intelligent Expert Finding and Team Building," 1/5/07 – 12/15/07, $50,000.
- Lockheed Martin, "ConX: Getting Answers on Demand," 8/1/07 – 1/31/08, $150,000.
- Office of Naval Research, "Rational Explanation of Beliefs, Intentions, and Threats (REBIT) Tool for Maritime Domain Awareness," 1/16/06 – 1/15/08, $627,736.
- Office of Naval Research, "Support for Predicting Improvised Explosive Device Attacks (SPIED)," 8/15/05 – 9/30/08, $900,000.
- Defense Advanced Research Projects Agency (DARPA), "Multi-Scale Behavioral Modeling and Analysis Promoting a Fundamental Understanding of Agent-based System Design and Operation," 6/1/00 – 9/30/06, Co-PIs: Luis Carpinella, Irene Gamba, Ari Arapostathis, $1,597,288.
- U.S. Army University XXI Program, "Sensible Agents for C3 Driver," 12/1/03 – 9/30/05, $200,000.
- U.S. Congress, "Systems Engineering and Distributed Agent-based Planning" National Consortium for Biological/Chemical Countermeasures administered through Institute of Advanced Technologies (IAT), Co-Investigators: Steve Kornguth (IAT), Gordon Boezer (Institute of Defense Analysis), Jim Chambers (UT-San Antonio), Andy Ellington (UT-Austin), George Georgiou (UT-Austin), Adam Heller (UT-Austin), Stephen Johnston (UT-Southwest Medical); Shelley Payne (UT-Austin), Michael Mastrangelo (BioD Communications), Dennis Perrotta (Texas Department of Health); Bob Shope (UT-Medical Branch), Eric Anslyn (UT-Austin), John McDevitt (UT-Austin), Mehmet Erengil (IAT), David Eaton (UT-Austin), Jerry Davis (UT-Center for Strategic Analysis), Steve Collier (State of

Texas, Office of Emergency Management), 6/1/04 – 10/30/05, $82,000, (Barber Portion), Total Program Amount: $1,150,000.

- Visitech and Defense Modeling and Simulation Office (DMSO), "Sensible Agents for Composable Simulations," 4/29/03 – 4/19/04, $75,000.
- ScenPro, Inc. and Air Force Research Laboratories, "Biological/Chemical Incident Response Monitor," 6/1/02 - 5/31/04, $167,479.
- IAT and U.S. Congress, "Information Systems for Distributed Chem-Bio Incident Response," 1/1/03 – 12/31/03, $120,000 (Barber portion), Multi-University, Multi-PI grant totaling $3.5 M.
- Defense Advanced Research Projects Agency (DARPA) and Army Research Laboratory (ARL), "Agents Augmenting Human Cognition for Measured Improved Performance, 3/1/02 – 09/30/04, $200,000.
- Schlumberger, "Management and Analysis of Software Engineering Artifacts to Promote Collaboration and Reuse," 7/1/01 – 8/30/06, $60,000.
- Defense Advanced Research Projects Agency (DARPA), "Distributed Sensible Agents Supporting Human-Information System Interaction: Dynamic Adaptive Autonomy For Responsive Decision-making," 1/1/00-5/17/03, $300,000.
- Defense Advanced Research Projects Agency (DARPA and National Cancer Institute), "Distributed Sensible Agents Supporting Human–Information System Interaction: Dynamic Adaptive Autonomy For Responsive Decision-making," 2/1/99 - 5/17/03, $1,752,131.
- Texas Higher Education Coordinating Board Advanced Technology Program, "Dynamic Configuration of Agent-based Decision-Making Systems for Manufacturing Enterprises," 1/1/00 – 12/31/01, $134,396. (Industry and Government Collaborators include: Harvard Manufacturing, MITRE, BBN/GTE Technologies, IC^2, Applied Research Laboratories, and Naval Surface Warfare Center)
- U.S. Congress, "Agent Systems: Review and Projections," University 21 Program administered through Institute of Advanced Technologies (IAT), Co-Investigators: Aubrey White (Institute of Advanced Technologies at UT-Austin), Bruce Porter (CS Department) at UT-Austin, Jim Wall (Texas A&M University) and John Stawasz (Applied Research Laboratories), 9/1/00 – 5/1/01, $50,000, (Barber Portion), Total Program Amount: $1,150,000.
- U.S. Congress, "Information Systems for Distributed Chem-Bio Incident Response," National Consortium for Biological/Chemical Countermeasures administered through Institute of Advanced Technologies (IAT), Co-Investigators: Steve Kornguth (IAT), Gordon Boezer (Institute of Defense Analysis), Jim Chambers (UT-San Antonio), Andy Ellington (UT-Austin), George Georgiou (UT-Austin), Adam Heller (UT-Austin), Stephen Johnston (Southwest Medical), Shelley Payne (UT-Austin); Michael Mastrangelo (Texas Department of Health), Dennis Perrotta (Texas Department of Health), Bob Shope (UT-Medical Branch); Ellen Wartella (UT-Austin), Eric Anslyn (UT-Austin), John McDevitt (UT-Austin), Mehmet Erengil (IAT), David Eaton (UT-Austin), Jerry Davis (UT-Center for Strategic Analysis), Steve Collier (State of Texas, Office of Emergency Management), 2/1/00 – 5/30/01, $120,551 (Barber Portion), Total Program Amount: $5M.
- Defense Modeling and Simulation Office (DMSO), "System Engineering for Advanced, Distributed Simulation Environments," 4/1/00 – 9/30/00, $60,000.
- U.S. Congress, "Distributed Decision-Making in the Digital Battlefield," University 21 Program administered through Institute of Advanced Technologies (IAT), Co-Investigators: Aubrey White (Institute of Advanced Technologies at UT-Austin), Bruce Porter (CS Department) at UT-Austin, Jim Wall (Texas A&M University) and John Stawasz (Applied Research Laboratories), 9/1/99 – 3/1/00, $50,000 (Barber Portion), Total Program Amount: $924,000.
- ScenPro, Inc. and Rome Laboratories, "Biological/Chemical Incident Response Monitor," Co-Investigators from Prime Contractor, ScenPro: Mark Swenholt and Lisa Mantock, 8/1/99 – 7/31/00, $30,797 (Barber Portion), Total Program Amount: $100,000.
- BBN Technologies/GTE and DARPA, "Architectures and System Configuration Evaluations of the Air Operations Enterprise Model," Co-Investigators from Prime Contractor, BBN Technologies: Marty Brown, 9/15/99 – 1/31/01, $871,793 (Barber Portion), Total Program Amount: $5.3M.
- Naval Surface Warfare Center, "Sensible Agents for Naval Command and Control," 11/1/99 – 5/1/00, $170,000.
- Army Research Laboratory, "Battlefield Digitization," 8/1/99 – 12/1/99, $50,000.
- Defense Advanced Research Projects Agency (DARPA and National Cancer Institute), "Distributed Sensible Agents Supporting Human-Information System Interaction: Dynamic Adaptive Autonomy For Responsive Decision-making," 2/1/99 – 7/31/00, $910,000.
- Allied Signal, "Software Engineering: Problems and Solutions," 4/15/98 – 8/31/00, $10,000.
- Applied Research Laboratory, "Java-based Simulation Frameworks," 1/15/98 – 5/31/98, $50,000.

- Defense Modeling and Simulation Office and Applied Research Laboratory, “Evaluation of Conceptual Modeling and Simulation Tool,” 12/10/97 – 8/31/98, $100,000.
- Defense Advanced Research Projects Agency (DARPA), “Knowledge Engineering Assistance and System Configuration Evaluation for the Protocol-driven Health Care Domains, 5/1/98 – 9/30/98, $160,519.
- Lockheed Martin, “Sensible Agent Planning for Conceptual Design Trade Studies,” 2/1/98 – 5/31/98, $50,000.
- Texas Higher Education Coordinating Board Advanced Technology Program, “Adapting the Autonomy of Sensible Agents: Coordinated Systems Operating in Complex, Dynamic Environments,” 1/1/97 – 8/31/00, $300,000. (Industry and Government Collaborators include: Lockheed Martin, Eastman Kodak, IC^2, Applied Research Laboratories, and Naval Surface Warfare Center).
- Defense Advanced Research Projects Agency (DARPA), “The Requirements and Integration Verification Tool (RIVT) Targeting Emergency Medical Care Applications,” 9/1/97 – 12/31/97, $99,959.
- Defense Advanced Research Projects Agency (DARPA), “Intelligent Agent Research & Development for Collaboration in Virtual Worlds used in Training: An Architecture to Support Varying Levels of Autonomy,” 7/15/97–8/31/97, $49,991
- Defense Advanced Research Projects Agency (DARPA), “Sensible Agent Research & Development in Support of Computer-Aided Education and Training Initiative Testbed Demonstrations,” 3/10/97 – 3/31/97, $52,500.
- Sloan Foundation, “Construction 21st Century,” Co Investigators: Richard L. Tucker, Jim T. O’Connor, Carl T. Haas, Jonathan F. Bard, Richard F. Brose, Alison Davis-Blake, G. Edward Gibson, J.G. Voeller, 1/15/97 – 12/31/99, $250,000 (Barber portion), Total Amount Awarded $1,980,000.
- Science Applications International Corporation (SAIC), “Tools and Technology for Affordable Multi-Missile Manufacturing,” 11/22/96 – 3/31/98, $235,536.
- Texas Higher Education Coordinating Board Advanced Technology Program, “Controlling the Autonomy of Sensible Agents in Manufacturing Environments,” Co-Investigator from UT-Arlington: Karan Harbison, 1/1/96 – 12/31/97, $169,400 (Barber Portion), Total Program Award: $294,400. (Industry and Government Collaborators include: Texas Instruments, Eastman Kodak, Tracor, Reveille Technology, IBM, and Naval Surface Warfare Center).
- Naval Surface Warfare Center, “Simulation-based Design of Combat Systems,” 8/15/95 – 2/1/97, $463,437 (Total Amount Received to Date: $71,200).
- Office of Naval Research, “Dynamic Adaptive Autonomy for Multi-Agent Systems,” 12/1/95 – 12/31/97, $130,000.
- L.L. and Ethel Dean Grants for Undergraduate Research, “A Visualization and Tracking System for a Virtual Decision Environment,” 5/1/95 – 5/31/96, $1,000.
- National Science Foundation (NSF), “Engineering Research Equipment: Virtual Decision Environment,” 3/1/95 – 2/29/96, $30,100.
- National Science Foundation (NSF), “The Stability of Autonomous Agents in Dynamic, Complex Manufacturing Environments,” 6/1/94 – 5/31/96, $49,999.
- Texas Higher Education Coordinating Board Advanced Technology Research, “Sensible Agents for Planning and Control in Dynamic Manufacturing Environments,” Co-Investigator from UT-Arlington: Karan Harbison, 1/1/94 – 8/31/96, $171,537 (Barber Portion), Total Program Award: $300,000. (Industry and Government Collaborators include: Techsus Medical Systems, National Center for Manufacturing Sciences, Quality Management Consortia, ScenPro, CTA, U.S. Air Force Wright-Patterson)
- Texas Higher Education Coordinating Board Advanced Technology Development, “Multi-Tool Integration Kernel for Component-based Architecture Development,” Co-Investigator from UT-Arlington: Karan Harbison, 1/1/94 – 8/31/96, $124,888 (Barber Portion), Total Program Award: $300,000. (Industry and Government Collaborators include: National Center for Manufacturing Sciences, IBM, Sarcos Research Corp., Texas Instruments, General Motors, Quality Management Consortia, U.S. Air Force Wright-Patterson)
- Vera Simons, “Virtual Reality for Intelligent Systems,” 6/9/93 – 6/8/96, $3,400.
- Texas Higher Education Coordinating Board Advanced Technology Research, Supplement to “Sensible Agents for Planning and Control in Dynamic Manufacturing Environments” grant to support minority undergraduate student , 11/1/94 – 5/1/95, $5,000.
- General Motors Foundation, “Intelligent Control,” 11/1/93 – 10/31/95, $20,000.
- Intel, Development of Virtual Environments, One Pentium-based computer system, Fall 1995, $2,000.
- Project QUEST “An Intuitive Interface for the Manipulation and Control of Virtual Environments,” One Macintosh 6100 and Laser Printer, Spring 1995, $3,000.

- IBM Equipment Grant, "Intelligent Product Life Cycle Workbench (PLCW)," PowerPC workstation, 1994, $4,000.
- Fluor Daniels, Inc., "Virtual Environments," 10/1/93 – 9/30/94, $4,000.
- Naval Surface Warfare Center, "The Identification of Research and Technology Appropriate to Naval Combat Systems," 6/1/93 – 7/31/93, $2,455.
- National Center for Manufacturing Sciences, "Intelligent Process Planning for Manufacturing Systems," 5/29/93 – 8/31/93, $9,950.
- The University Research Institute, "Fuzzy, Agent-based Resource Specification Incorporating Product and Process Knowledge for a Manufacturing Facility" 6/1/93 – 7/31/93, $12,556.
- Martin Marietta, Inc., "Express Schema Development of the Information Base for Next Generation Controller Program," Co-Investigator: K.A. Harbison-Briggs, 8/1/91–1/15/92, $16,500.
- Martin Marietta, Inc., "Specification for the Information Base for the Next Generation Controller Program," Co-Investigator: K.A. Harbison-Briggs, 8/1/91 – 1/15/92, $175,000.
- Martin Marietta, Inc., "Requirements Analysis and Specification for the Information Base for the Next Generation Controller Program," Co-Investigator: K.A. Harbison-Briggs, 2/7/91 – 2/10/92, $46,000.
- Martin Marietta, Inc., "Evaluation, Testing, and Compliance for Platform Services in the Open Architecture for the Next Generation Controller Program," Co-Investigators: K.A. Harbison-Briggs and J.C.M. Tiernan, 1/5/91 – 3/30/91, $60,000.
- Automation and Robotics Research Institute (ARRI), "Product Data Driven Controller (PDDC)," 12/1/88 – 12/1/91, $150,000.

ADMINISTRATIVE AND COMMITTEE SERVICE:

Department Committees

- ECE Community Well-being and Support, Co-Chair, 2020 – present.
- Software Engineering Graduate Students Admissions Committee, 2004 – present.
- Software Engineering Prequalifying Committee, 2010 – present.
- Faculty Advisor, Electrical and Computer Engineering, September 1992 – present.
- Senior Faculty Search Committee, September 2016 – August 2017.
- Faculty Evaluation Committee, September 2015 – August 2016.
- Faculty Advisor, Software Engineering Executive M.S. (Option III) Degree Program, September 2002 – August 2015.
- ECE Faculty Hiring Committee, 2012-2013.
- Department Faculty Evaluation Committee, Fall 2011.
- Software Engineering Faculty Recruiting Committee, September 2007 – 2011
- ECE Graduate Student Recruiting Committee, 1992 – 2010.
- Software Engineering Area Committee, Chair, 2005 – 2010.
- Departmental Faculty Strategic Hiring Committee, Fall 2004 – 2009.
- Software Engineering Undergraduate Curriculum Committee, Chair, 2002 – 2008.
- Management and Production Course Area Committee, 1992 – 2005.
- Control Systems Area Committee, 1993 – 2004.
- Industry Liaison Committee, 1993 – 2004
- Minority Student Committee, 1992 – 2004.
- Freshman Course Area Committee, Chair: Spring 1997 – Fall 1998; member: Fall 1998-2003.
- Software Engineering Lecture Series, Organizing Committee, Summer-Fall 1998; Summer 1999 – Spring 1999.
- ENS Renovations Committee, Fall 1997 – Fall 1999.
- ECE Chair Search Committee, Spring 1997.
- Adhoc Committee to Review the Undergraduate Curriculum, 1994 – 1995.
- Software Design and Development (M.S. curriculum development), joint committee with Computer Sciences, 1993–1994.

- Circuits and Systems Course Area Committee, 1992 – 1994.

School of Engineering Committees

- Cultural Awareness Committee, Faculty Chair, 2020 - present.
- Diversity, Equity and Inclusion Committee, 2020 – present.
- Engineering Equal Opportunity Advisory Committee, 2011 – 2016.
- Schlumberger Center of Excellence Team, 2001 – 2012.
- Hocott Award Committee, 1996, 1999, 2002, 2006, 2007, 2008.
- Women in Engineering Program Steering Committee, 1995 – Fall 2005.
- Manufacturing Steering Committee, 1994 – 2002.
- Review Committee for the Executive Software Engineering Program, Spring 1999.
- Ad Hoc Committee for Center in Manufacturing, August 1996 – 1999.
- Systems and Software Engineering Institute, Chairperson of the Board, August 1996 – September 1999.
- Consultative Committee for Director-Manufacturing Systems Engineering, June 1996 – 1998.
- Systems and Software Engineering Institute, Board Chair (1996 – present) and Program Committee (1995 –1999).

University Committees

- Center for Identity, Director, September 2010 – Present.
- M.S. degree program in Information Security and Privacy, Director & Faculty Advisor, September 2015 – present.
- Child Identity Protection Task Force, Center for Identity, Spring 2012 – 2016.
- Department Chair and Center Director Leadership Working Group, Spring 2005.
- Curricular initiative for Information Technology, Academic Year 2000-2001.
- Consultative Committee for Selection of The University of Texas VP-Research, September 1998 – May 1999.
- Ad-Hoc Search Committee for Information Technology Division Director, Applied Research Labs, April –May 1998.
- Intellectual Property Rights Committee, 1997 – 2000.
- Committee for the Support and Advancement of Women, Advisory Committee to the President, Fall 1997 – 2000.
- Research Infrastructure Enhancement Committee (RISE), 1995 – 2001.
- Women Leading Technology, Steering Committee, 1996 – 2001.
- Parking and Traffic Committee, 1997 – 1999.
- Committee on The University of Texas Core Values and Purpose, 1996 – 1997.

Government Committees

- Department of Homeland Security (DHS) Data Privacy and Integrity Advisory Committee, Subcommittee on Biometric Technology and Policy serving DHS Secretary, 2017 – 2019.
- Department of Homeland Security (DHS) Data Privacy and Integrity Advisory Committee serving the Secretary of Homeland Security, June 2012 – December 2019.
- National Aviation Intelligence Integration Office, Insider Threat Advisory Committee, 2016 – 2019.
- Information Security Workforce Development Committee, commissioned by Department of Information Resources, State of Texas, 2013 – 2016.
- Army Science Board, September 2005 – 2016.
- Texas First Responder Advisory Committee (FRAC), September 2011 – 2016.
- U.S. Information Sharing Environment (http://ise.gov/), August 2011 – 2016.
- Texas Homeland Security IT Subcommittee on Identity, May 2011 – 2016.
- ANSI, Protected Health Information (PHI) Advisory Committee, April 2011 – September 2012.

- Institute of Defense Analysis Advisors and Consultants, September 2004 – August 2012.
- DARPA Intelligent Processing Technology Office Working Group, 2005 – 2006.
- Defense Science Study Group, DARPA and Institute of Defense Analysis (IDA), 2002 –2003.
- Engineering for Design and Manufacturing Panel, Review of National Science and Technology Board Manufacturing Infrastructure Subcommittee directions, September 1994 – 2000.
- National Research Council, Site Visitor and Review Panelist, June 1994 –September 2002.
- Texas Higher Education Coordinating Board Advanced Research Program and Advanced Technology Program, Panel member reviewing ATP program, December 8, 1994.
- Computer Science and Telecommunications Board, Manufacturing Studies Board, National Research Council, March 1994 – September 1994.
- Intelligent Product Processing Initiative (IPPI) sponsored by U.S. Air Force, Technical Review Board, 1994 – 1995.
- Next Generation Controller (NGC) Program sponsored by the U.S. Air Force and National Center for Manufacturing Sciences, Technical Review Board, September 1990 to August 1994.
- Air Force Wright Laboratory Intelligent Machining Workstation (IMW) Program, Industrial Review Board, 1988 –1989.

Conference Committees

- International Conference on Intelligent Systems and Applications (INTELLI 2020), June 28 - July 02, 2020, Athens, Greece, Program Committee.
- International Conference on Agents and Artificial Intelligence (ICAART), February 22-24, 2020, Valleta, Malta, Program Committee.
- International Conference on Agents and Artificial Intelligence (ICAART), February 19-21, 2019, Prague, Czech Republic, Program Committee.
- International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, July 9-11, 2019, Graz, Austria, Program Committee.
- International Conference on Intelligent Systems and Applications (INTELLI 2019), June 30 - July 04, 2019, Rome, Italy, Technical Review Committee.
- Artificial Intelligence International Conference, November 21-23, 2018, Barcelona, Spain. Review Committee.
- International Conference on Agents and Artificial Intelligence (INTELLI 2018), June 24 - 28, 2018, Venice, Italy, Technical Review Committee.
- International Conference on Agents and Artificial Intelligence (ICAART), January 16-18, 2018, Funchal-Madeira, Portugal, International Program Committee.
- International Conference on Agents and Artificial Intelligence (ICAART), February 24-26, 2017, Porto, Portugal, International Program Committee.
- International Conference on Industrial, Engineering & Other Applications of Applied Intelligent System, June 27-30, 2017, Arras, France, Program Committee.
- IEEE International Conference on Privacy, Security and Trust (PST), December 12-14, 2016, Auckland, New Zealand, International Program Committee.
- Global Identity Summit, September 19-24, 2016, Program Planning Committee.
- Global Identity Summit STEM Showcase, September 19-24, 2016, Organizing Committee, Chair.
- International Conference on Industrial, Engineering & Other Applications of Applied Intelligent System, August 2-4, 2016, Morioka, Japan, Program Committee.
- AAAI Conference on Artificial Intelligence (AAAI), 5th AAAI Workshop on Incentives and Trust in E-Communities (WIT-EC) 2016, February 12–17 2016, Phoenix, Arizona, International Program Committee.
- International Conference on Agents and Artificial Intelligence (ICAART), February 24-26, 2016, Rome Italy, International Program Committee.
- Global Identity Summit, September 14-16, 2015, Tampa, FL, Planning Committee.
- 28th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent System, June 10-12, 2015, Seoul, Korea, Program Committee.

- 4th Workshop on Incentive and Trust in Electronic Communities, 29th AAAI Conference on Artificial Intelligence (AAAI), January 25–29, 2015, Austin Texas, Program Committee.
- International Conference on Agents and Artificial Intelligence (ICAART), January 10-12, 2015, Lisbon, Portugal, International Program Committee.
- IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'14), August 11-14, 2014, Warsaw, Poland, Program Committee.
- ID360: Global Forum on Identity, April 9-10, 2014, Austin, TX, Chair, Organizing Committee.
- International Conference on Agents and Artificial Intelligence (ICAART), March 6-8, 2014, Angers, France, International Program Committee.
- International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, June 17-21, 2013, The Netherlands, Program Committee.
- ID360: Global Forum on Identity, April 30 and May 1, 2013, Austin, TX, Chair, Organizing Committee.
- Trust Workshop, International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2013), May 7, 2013, Minneapolis, MN, Organizing Committee.
- ID360: Global Forum on Identity, April 23-24, 2012, Austin, TX, Chair, Organizing Committee.
- Trust Workshop, International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS), June 4-8, 2012, Valencia, Spain, Organizing Committee.
- International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS), June 4-8, 2012, Valencia, Spain, Program Committee.
- International Conference on Agents and Artificial Intelligence, February 6-8, 2011, Vilamoura - Algarve, Portugal, Program Committee.
- International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2011), Trust in Agent Societies Workshop, May 2, 2011, Taipei, Taiwan, Program Committee.
- International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2010), May 10, 2010, Toronto, Canada, Program Committee.
- Co-organizer and Program Committee, "Reputation and Trust Workshop, International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2010), Toronto, Canada, May 10, 2010, Program Committee.
- International Conference on Agents and Artificial Intelligence, Valencia, Spain, January 22-24, 2010, Program Committee.
- Third International Workshop on Massively Multi-Agent Systems: Models, Methods and Tools (MMAS'09), Budapest, Hungary, May 11 2009, Program Committee.
- Trust in Agent Societies Workshop, Autonomous Agents & Multi-Agent Systems (AAMAS 2009), Budapest, Hungary, May 11 2009, Co-organizer and Program Committee.
- International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2007), Lisbon, Portugal, Program Committee.
- Trust, Fraud and Deception Workshop, International Joint Conference on Autonomous Agents & Multi-Agent Systems Honolulu, Hawaii; 2007, Co-Organizer and Program Committee.
- IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, (WI/IAT 2006), Hong Kong, China, December 2006, Program Committee.
- Programming Multi-Agent Systems (ProMAS), International Joint Conference on Autonomous Agents & Multi-Agent Systems, Hakodate, Japan, May 8-12, 2006, Program Committee.
- Defense Applications and Multi-Agent Systems Workshop, International Joint Conference on Autonomous Agents & Multi-Agent Systems, Hakodate, Japan, May 8-12, 2006, Program Committee.
- Trust, Fraud and Deception Workshop, International Joint Conference on Autonomous Agents & Multi-Agent Systems Hakodate, Japan, May 8-12, 2006, Co-Organizer and Program Committee.
- International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2006), Hakodate, Japan; May 8-12, 2006, Program Committee.
- IEEE Symposium on Multi-Agent Security and Survivability (MAS&S), Philadelphia, PA, August 30-31, 2005, Co-Chair.
- IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology Intelligent Agent Technology, (WI/IAT 2005), Compiègne University of Technology, France, September 19-22, 2005, Program Committee.

- Defense Applications and Multi-Agent Systems Workshop, International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2005), Utrecht, the Netherlands, July 25-29, 2005, Program Committee.
- Trust, Fraud and Deception Workshop, International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2005), Utrecht, the Netherlands, July 25-29, 2005, Co-Organizer and Program Committee.
- International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2005), Utrecht, the Netherlands, July 25-29, 2005, Program Committee.
- IEEE Symposium on Multi-Agent Security and Survivability (MAS&S), Philadelphia, PA, August 30-31, 2004, Program Committee.
- IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004), Beijing, China, September 20-24, 2004, Program Committee.
- Workshop on Trust, Privacy, Deception and Fraud in Agent Societies, International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2004), New York City, New York, July 2004, Co-Organizer.
- Workshop on Multiagent Planning and Scheduling, International Conference on Automated Planning and Scheduling, Monterey, CA, June 6, 2005, Program Committee.
- International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, July 2003, Program Committee.
- International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2003), Melbourne, Australia, July 2003, Program Committee.
- Workshop on Autonomy, Delegation, and Control: From Inter-agent to Organizations and Institutions, International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2003), Melbourne, Australia, July 2003, Program Committee.
- International Workshop on Computational Autonomy (AUTONOMY 2003), Melbourne, Australia, July 2003, Program Committee.
- Workshop on Resource, Role, and Task Allocation in Multi-Agent Systems, International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2003), Melbourne, Australia, July 2003, Program Committee.
- Workshop on Trust, Privacy, Deception and Fraud in Agent Societies, International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2003), Melbourne, Australia, July 2003, Program Committee.
- RoboCup 2002 Conference, Fukuoka, Japan and Busan, South Korea, together with the RoboCup soccer and rescue competitions, June 19-25, 2002, International Program Committee.
- International Workshop on Agents in Design (WAID2002), MIT, Cambridge, August 28-30 2002, Program Committee.
- Eighteenth National Conference on Artificial Intelligence (AAAI-2002), Workshop on Autonomy, Delegation, and Control: From Inter-agent to Groups, Edmonton, Alberta, Canada, July 28, 2002, Program Committee.
- International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2002), Workshop on Toward an Application Science: MAS Problem Spaces and Their Implications, Bologna, Italy from July 15 - 19, 2002, Organizing Committee.
- International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2002), Workshop on Deception, Fraud and Trust in Agent Societies, Bologna, Italy from July 15 - 19, 2002, Organizing Committee.
- International Joint Conference on Autonomous Agents & Multi-Agent Systems - AAMAS 2002 (formerly AGENTS and ICMAS), Bologna, Italy from July 15 - 19, 2002, Program Committee.
- International Joint Conference on Artificial Intelligence (IJCAI 2001), Workshop on Artificial Intelligence and Manufacturing: New AI Paradigms for Manufacturing, Seattle, WA, August 5, 2001, Organizing Committee.
- International Joint Conference on Artificial Intelligence (IJCAI 2001), Autonomy, Delegation, and Control Workshop, Seattle, WA, August 2001, Program Committee.
- Second Asia-Pacific Conference on Intelligent Agent Technology (IAT'2001), Maebashi City, Japan, October 23-26, 2001, Program Committee.
- International Workshop on Agent Theories, Architectures, and Languages (ATAL-01), Seattle, WA, August 1-3, 2001, Program Committee.
- International Workshop for Fraud, Deception and Trust in Agent Societies, Autonomous Agents 2001, May 28, 2001, Montreal, Canada, Organizing Committee and Program Committee.
- International Conference on Autonomous Agents 2001, May 28-June 1, 2001, Montreal, Canada, Program Committee.

- International Workshop on Autonomy Oriented Computation (AOC), May 28, 2001, Montreal, Canada, Organizing Committee and Program Committee.
- International Workshop on Infrastructure for Agents, MAS, and Scalable Multi-Agent Systems, Autonomous Agents 2001, May 28, 2001, Montreal, Canada, Organizing Committee and Program Committee.
- The Fourteenth International Conference on Florida Artificial Intelligence Research Symposium (FLAIRS'01), Key West, FL., May 20–23, 2001, Program Committee.
- The Eleventh IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 2000), December 2000, Technical Program Committee and Session Chair.
- The Sixth International Conference on Intelligent Autonomous, Special Track for Multi-Agent Architectures for Intelligent Automation, Venice, Italy, July 25–27, 2000, Organizing Co-Chair.
- Pacific Rim International Conference on Artificial Intelligence (PRICAI), Workshop on Teams with Adjustable Autonomy, Melbourne, Australia, August 28–29, 2000, Organizing Committee.
- The Seventeenth National Conference on Artificial Intelligence (AAAI–2000), Intelligent Systems Demonstrations, Austin, TX., July 30 – August 3, 2000, Organizing Committee.
- The Fourth International Conference on Autonomous Agents (Agents 2000), Infrastructure for Research-Grade MAS Construction Workshop, Barcelona, Catalonia, Spain, June 3 – 7, 2000, Organizing Committee.
- The Twelfth International Conference on Software Engineering and Knowledge Engineering (SEKE2000), Chicago, IL, July 6–8, 2000, Program Committee.
- The Seventh International Workshop on Agent Theories, Architectures, and Languages (ATAL-2000), Boston, MA., July 7–9, 2000, Program Committee.
- The Fourth International Conference on Multi-Agent Systems, Infrastructure for Research-Grade MAS Construction Workshop, July 7–12, 2000, Boston MA., USA, Organizing Committee.
- The Fourth International Conference on Autonomous Agents (Agents 2000), Barcelona, Catalonia, Spain, June 3 – 7, 2000, Program Committee.
- The Twenty-Second International Conference on Software Engineering (ICSE), Intelligent Software Engineering (ISE3) Workshop, Limerick, Ireland, June 4–11, 2000, Organizing Committee.
- The Thirteenth International Conference on Florida Artificial Intelligence Research Symposium (FLAIRS'00), Orlando, FL., May 2000, Program Committee.
- The Twelfth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE99), Workshop on Intelligent Agents for Electronic Commerce, May 31 – June 3, 1999, Cairo, Egypt, Program Committee.
- The Twelfth Workshop on Knowledge Acquisition, Modeling and Management, KA for Requirements Engineering Track, October 16-21, 1999, Co-Chair.
- The Twelfth International Conference on Florida Artificial Intelligence Research Symposium (FLAIRS'99), Orlando, FL., May 1–5, 1999, Program Committee.
- ISADS99, The Fourth International Symposium on Autonomous Decentralized Systems, Emergent Approach session, Tokyo, Japan, March 21–23, 1999, Session Chair.
- International Conference on Agile, Intelligent and Computer-Integrated Manufacturing, Troy, NY, October 7–9, 1998, International Program Committee.
- Agent-Based Manufacturing Workshop, Autonomous Agents '98, Minneapolis, MN, May 9, 1998, Program Committee.
- AI and Manufacturing: State-of-the Art and State-of-the-Practice, AAAI SIGMAN (Special Interest Group in Manufacturing) Conference, Albuquerque, NM., August 31 – September 2, 1998, Program Organizing Committee.
- The Eleventh International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE-98), Benicassim, Castellon, Spain, June 1–4, 1998, Invited Session Organizer.
- 1997 IEEE International Conference on Robotics and Automation, Albuquerque, NM, April 20–25, 1997, Program Committee.
- Intelligent Systems Conference, Workshop on Semiotic Modeling for Sensible Agents, Gaithersburg, MD, October 23–25, 1996, Workshop Co-Chair (Co-Chair: Srini Ramaswamy).
- AAAI SIGMAN (American Association for Artificial Intelligence Special Interest Group in Manufacturing) Research Planning Workshop, Albuquerque, NM, April 24–26, 1996, Working Group Chair.

- National Institute of Standards and Technology Manufacturing Process Planning Workshop and CAME (Computer Aided Manufacturing Engineering) Forum Workshop, Gaithersburg, MD, June 10–11, 1996, Session Chair.
- The Second Internal Federation for Information Processing (IFIP 5.10) Workshop on Virtual Prototyping, "CAD and Virtual Prototyping," May 6, 1996, Panel Member.
- The Fourth National Conference on "Diversity in the Scientific and Technological Workforce," sponsored by The National Science Foundation, Electrical, Computer, and Solid-State Engineering, Washington, D.C., September 22, 1995, Panel Co–Facilitator (Co-Facilitator: Carmen Menoni).
- 1995 IEEE International Symposium on Intelligent Control (ISIC), Monterey, CA., August 27–29, 1995, Co-Program Chair.
- 1994 IEEE International Conference on Robotics and Automation, San Diego, CA., May 17–21, 1994, Technical Committee.
- 1994 IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, October 2–5, 1994, Session Chair.
- Workshop on Education for Manufacturing 2002 sponsored by UT College of Engineering Manufacturing Steering Committee and Gulf-Southwest Section of American Society for Engineering Education, Austin, TX, May 12, 1994, Program Committee.
- The Automated Factory of the Future: Where Do We Go From Here? Workshop at the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA., May 9, 1994, Co-Chairperson (Co-Chairperson: Dr. Eric Mettala).
- Intelligent Manufacturing Workshop, American Association for Artificial Intelligence Conference conducted July 11, 1993, Invited Program Committee.

External Advisory Committees and Boards

- Advisory Board, Computer Science Department, Concordia University, 2013 – present.
- Privacy and Security Committee, Pecan Street Consortium, Co-Chair.
- Metroplex Technology Business Council, 2008 – 2012, Cloud Security Committee.
- Center for Applied Identity Management Research (CAIMR), 2008 – 2010, Research Planning Committee.
- Center for Applied Identity Management Research (CAIMR), 2008 – 2010, Membership Committee.
- SEMATECH, 1997 – 2000, Factory Integration Technical Working Group.
- National Center for Manufacturing Sciences, 1993 – 1995, Factory Controls Tactical Action Committee.

Professional Society Committees

- IEEE Technical Council on Software Engineering, 1995 – present.
- American Association for Artificial Intelligence Special Interest Group in Manufacturing, Academic Co-Chair.

Editorial Boards

- International Journal of Agent Technologies and Systems, Editorial Board, 2010 – present.
- Journal of Database Management, Editorial Review Board, 2001- present.
- Journal of Experimental and Theoretical Artificial Intelligence, Special Issue: Agent-based Systems and Autonomy Control Software, 1999.
- Journal of Intelligent Automation and Soft Computing (IASC), Special Issue Editor, 1996.

PROFESSIONAL SERVICE ACTIVITIES:

Book Reviews

- Review for book titled "Trust in E-Services: Technologies, Practices and Challenges," publishers Idea Group, 2006.
- Review for the 2000 PRICAI Post-proceedings, Springer LNAI volume, Reviewer 2001.
- Review for Pearson Education (the company formed as a result of the merger between Addison-Wesley Longman and Prentice Hall) of the book titled Compositional Design of Intelligent Agent Systems by Frances Brazier, Catholijn Jonker and Jan Treur, 2001.

- PRICAI Workshop Reader of Agent-based Software Teams with Adjustable Autonomy, Post-Proceedings of the Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000), Springer LNAI volume, 2001.

Journal Reviews

- International Journal of Information Privacy, Security and Integrity (IJIPSI), Reviewer, 2019 – present.
- ACM Transactions on Intelligent Systems and Technology, Guest Editor in 2014, 2010 – present.
- Journal of Artificial Intelligence Research, 2010 – present.
- Journal of Applied Artificial Intelligence, Reviewer, 2003 – present.
- Journal of Autonomous Agents and Multi-Agent Systems, Reviewer, 2002 – present.
- IEEE Systems, Man and Cybernetics, 2002 – present.
- IEEE Computer, Review, 2001 - present.
- Journal of Intelligent Information Systems, Reviewer, 1997 - present.
- ACM Transactions on Internet Technology, 2015.
- Encyclopedia of Software Engineering, 2010 – 2016.
- Knowledge Engineering Review, Reviewer, 2005 – 2016.
- International Journal on Integrated Computer-Aided Engineering, Reviewer, 2005 – 2016.
- Journal of Software Maintenance and Evolution, Reviewer, 2005 – 2016.
- Journal of Software – Practice and Experience, Reviewer, 2002 – 2016.
- International Journal on Knowledge and Information Systems (KAIS), Reviewer, 2001 – 2016.
- IEEE Transactions on Software Engineering, Reviewer, 2001 – 2016.
- Journal of Production Planning and Control, Reviewer, 1998 - 2016
- Journal of Manufacturing Systems, Reviewer, 1997 - 2016.
- Requirements Engineering Journal, Reviewer, 2002.
- Journal of Artificial Intelligence in Engineering Design and Manufacturing (AIEDAM), Special Issue: on New AI Paradigms for Manufacturing, Reviewer, 2002.
- Journal of Integrated Computer-Aided Engineering, Reviewer, 2002, 1998 - 1999
- IEEE Transactions on Knowledge and Data Engineering (TKDE), Reviewer, 2001.
- International Journal on Artificial Intelligence Tools (IJAIT), Reviewer, December 2000 Special Issue.
- IEEE Transactions on Automation and Robotics, Reviewer, 1997.
- Journal of Engineering Education, 1996.

MEMBERSHIPS IN PROFESSIONAL AND HONORARY SOCIETIES:

- Institute of Electrical and Electronic Engineers, (Computer Society, Robotics & Automation Society, and Systems, Man, and Cybernetics Society), Senior member.
- IEEE Technical Council on Software Engineering (TCSE), member.
- IEEE Computer Society Golden Core, Senior member.
- American Association for Artificial Intelligence, member.
- Association for Computing Machinery (ACM), member.
- International Society of Applied Intelligence, member.
- American Society of Engineering Education, member.
- Foundation for Intelligent Physical Agents, member.
- Society of Computer Simulation, past member.
- Society of Women Engineers, member.
- Sigma Xi, member.
- Phi Kappa Phi, member.

APPENDIX B

Appendix B

1. U.S. Patent No. 6,411,941 ANCORA_00000481-ANCORA_00000490
2. Prosecution History of U.S. Patent No. 6,411,941
3. Prosecution History of '941 patent with ex parte reexamination, ANCORA_00003077-ANCORA_00003333
4. Prosecution History for Ex Parte Reexamination of U.S. Patent No. 6,411,941, filed May 28, 2009, Control No. 90/010,560 ANCORA_00003077-ANCORA_00003333
5. Federal Circuit Opinion from *Ancora Technologies Inc. v. Apple Inc.*, 744 F.3d 732 (2014)
6. Federal Circuit Opinion from *Ancora Technologies Inc. v. HTC America Inc.*, 908 F.3d 1343 (2018)
7. "Joint Claim Construction Statement" of the parties, dated May 1, 2020 (ECF No. 56)
8. "Final Claim Constructions of the Court," dated June 2, 2020 (ECF No. 69)
9. "Supplemental Claim Construction Order," dated August 19, 2020 (ECF No. 93)
10. Ancora's Final Infringement Contentions, October 2, 2020
11. Defendants' Final Invalidity Contentions, October 2, 2020
12. U.S. 4,658,093 ("Hellman") DEFS_ANCORA000001-DEFS_ANCORA000012
13. U.S. Patent No. 5,935,243 ("Hasebe")
14. U.S. Patent No. 6,138,236 ("Mirov") DEFS_ANCORA00000435-DEFS_ANCORA00000444
15. U.S. Patent No. 4,757,534 ("Matyas") DEFS_ANCORA00001206-DEFS_ANCORA00001228
16. U.S. Patent No. 5,802,592 ("Chess") DEFS_ANCORA00000887-DEFS_ANCORA00000894
17. U.S. Patent No. 6,243,468 ("Pearce") DEFS_ANCORA00000552-DEFS_ANCORA00000566
18. U.S. Patent No. 5,852,736 ("Shipman")
19. Desktop Management BIOS Specification Version 2.0, March 6, 1996 ("DMI Spec.") DEFS_ANCORA00000831-DEFS_ANCORA00000879
20. U.S. Patent No. 5,892,906 ("Chou") DEFS_ANCORA00000323-DEFS_ANCORA00000334
21. U.S. Patent No. 5,421,006 ("Jablon") DEFS_ANCORA00000917-DEFS_ANCORA00000942
22. U.S. Patent No. 6,153,835 ("Schwartz") DEFS_ANCORA00000468-DEFS_ANCORA00000498

23. U.S. Patent No. 6,185,678 (“Arbaugh”) DEFS_ANCORA00000499-DEFS_ANCORA00000524
24. Ph.D. Thesis of Bennett Yee, “Using Secure Coprocessors”, Carnegie-Mellon University, CMU-CS-94-149 (“Yee”) DEFS_ANCORA000695-DEFS_ANCORA000798
25. U.S. Patent No. 5,892,900 (“Ginter”)
26. U.S. Patent No. 5,748,804 (“Isikoff”) DEFS_ANCORA00001185-DEFS_ANCORA00001197
27. B. Schneier, Applied Cryptography, Second Edition (1996) (“Schneier”)
28. W.R. Cheswick et al., Firewalls and Internet Security (1994) (“Cheswick”)
29. Intel-28F001BX-B-datasheet (“Intel”) DEFS_ANCORA00001136-DEFS_ANCORA00001169
30. U.S. Patent No. 6,009,524 (“Olarig”) DEFS_ANCORA00001198-DEFS_ANCORA00001205
31. U.S. Patent No. 5,684,951 (“Goldman”)
32. U.S. Patent No. 6,189,146 (“Misra”) DEFS_ANCORA00000525-DEFS_ANCORA00000545
33. U.S. Patent No. 5,479,639 (“Ewertz”)
34. U.S. Patent No. 6,009,177 (“Sudia”)
35. U.S. Patent No 5,473,692 (“Davis”) DEFS_ANCORA_00000129-DEFS_ANCORA_00000144
36. U.S. Patent No. 5,568,552 (“Davis”) DEFS_ANCORA_00000222-DEFS_ANCORA_00000237
37. U.S. Patent No. 5,666,411 (“McCarty”) DEFS_ANCORA_00000256-DEFS_ANCORA_00000280
38. U.S. Patent No. 6,523,119 (“Pavlin”) DEFS_ANCORA_00000607-DEFS_ANCORA_00000623
39. Silberschatz, Operating System Concepts, 5th edition (1997) (“Silberschatz”)
40. Microsoft Computer Dictionary, 3rd edition (1997) (“Microsoft”)
41. Israel Application No. 124,571, filed May 21, 1998 (“the IL’571 application”) ANCORA_00007080- ANCORA_00007105
42. RSA, “Public Key Cryptography Standard (PKCS) #7: Cryptographic Message Syntax Standard” (1993)
43. IETF RFC 951, “Bootstrap Protocol,” September 1985
44. Secure Hash Standard, Technical Report FIPS 180-1, U.S. Department of Commerce, April 1995

45. Digital Signature Standards, Technical Report FIPS- 186, U.S. Department of Commerce, May 1994 (“DSS”)
46. Ancora’s Second Supplemental Responses and Objections to LGE’s First Set of Interrogatories (December 23, 2020)
47. Expert Report of Dr. Scott Nettles and Exhibits thereto
48. Expert Report and Declaration of Dr. Arbaugh and Exhibits thereto
49. Expert Report and Declaration of Dr. Hicks and Exhibits thereto
50. Expert report of Dr. Hassoun and Exhibits thereto
51. Rebuttal Expert Report of Dr. Martin
52. Transcript of Inventor Deposition of Miki Mullor in Ancora v. Apple (Oct. 15, 2015) ANCORA_00047721-ANCORA_00048028
53. Transcript of 30(b)(6) Deposition of Miki Mullor in Ancora v. Apple (Dec. 22, 2015) ANCORA_00048029-ANCORA_00048199
54. Transcript of Deposition of Miki Mullor in Ancora v. Samsung (Nov. 2, 2020) and Exhibits Thereto
55. Transcript of the Markman hearing from *Ancora v Apple* (NDCA) matter, dated June 29, 2012
56. *HTC Corp. v. Ancora Tech., Inc*., Decision Denying Institution of Covered Business Method Patent Review CBM2017-00054 (Dec. 1, 2017)
57. University of Pennsylvania Department of Computer and Information Science as Technical Report No. MS-CIS-96-35 (“Arbaugh 1996”) DEFS_ANCORA00000821-DEFS_ANCORA00000830
58. “A Secure and Reliable Bootstrap Architecture,” published in the Proceedings of the 1997 IEEE Symposium on Security and Privacy (“Arbaugh IEEE 1997”) DEFS_ANCORA00000880-DEFS_ANCORA00000886
59. “Automated Recovery in a Secure Bootstrap Process,” University of Pennsylvania Department of Computer and Information Science as Technical Report No. MS-CIS-97-13. (“Arbaugh Tech. Report 1997”) DEFS_ANCORA00000970-DEFS_ANCORA00000988
60. SwitchWare Papers: D.S. Alexander et al., “The SwitchWare Active Network Architecture”, IEEE Network, May/June 1998, pp. 29-36 DEFS_ANCORA00003251 - DEFS_ANCORA00003258
61. SwitchWare Papers: D. S. Alexander et al., “A Secure and Active Network Environment Architecture: Realization in SwitchWare”, IEEE Network, May/June 1998, pp. 37-45 DEFS_ACNORA00003086- DEFS_ACNORA00003098
62. SwitchWare Papers: D.S. Alexander et al., “Secure Quality of Service Handling: SQoSH”, IEEE Communications Magazine, April 2000, pp. 106-112 DEFS_ANCORA00003283- DEFS_ANCORA00003289

63. SwitchWare Papers: D.S. Alexander et al., “Safety and Security of Programmable Network Infrastructures”, IEEE Communications Magazine, October 1998, pp. 84-92 DEFS_ANCORA00003312-DEFS_ANCORA00003316
64. Arbaugh Lab Notebook 1 DEFS_ANCORA00002325-DEFS_ANCORA00002480
65. Arbaugh Lab Notebook 2 DEFS_ANCORA00002481-DEFS_ANCORA00002728
66. Arbaugh Lab Notebook 3 DEFS_ANCORA00002729-DEFS_ANCORA00002925
67. Additional Arbaugh Materials DEFS_ANCORA00003592-DEFS_ANCORA00003619
68. Arbaugh file history and provisional DEFS_ANCORA00003363-DEFS_ANCORA00003530
69. Arbaugh provisional U.S. Pat. App. No. 60/060,885 DEFS_ANCORA00003531-DEFS_ANCORA00003581
70. Arbaugh: Thesis proposal email message DEFS_ANCORA00002926-DEFS_ANCORA00002927
71. Beeble email to Microsoft employee re patent ANCORA_00003911-ANCORA_00003918
72. See Intel 1-MBIT (128K x 8) BOOT BLOCK FLASH MEMORY, 28F001BX-T/28F001BX-B/28F001BN-T/28F001BN-B (1995) DEFS_ANCORA00001136-DEFS_ANCORA00001169
73. Samsung Inter Petition for Inter Partes Review of US 6,411,911 filed by Samsung Defendants and supporting exhibits, including the Declaration of Dr. Erez Zadok [PTAB Case No. IPR2020-01184]
74. Patent Owner’s Preliminary Response (“POPR”) in the ’941 Patent IPR [PTAB Case No. IPR2020-01184]
75. IETF RFC 951: Bootstrap protocol (BOOTP, September 1985)
76. 2020-09-10 IPR2020-01609 Petition for *Inter Partes Review* of US 6,411,941
77. Declaration of Andrew Wolfe, Ph,D, (“Wolfe Decl.”), 09/10/2020, from IPR2020-01609, Ex. 1003
78. Plaintiff’s Markman Technology Tutorial
79. Defendants’ Markman Technology Tutorial
80. U.S. Patent No. 5,822,581 (“Christeson”) DEFS_ANCORA00003350-DEFS_ANCORA00003362