

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

APPLE INC.
Petitioner,

v.

CONTENTGUARD HOLDINGS, LLC
Patent Owner.

U.S. Patent Nos. 6,963,859; 7,523,072; 8,370,956; 8,393,007; 7,269,576

Inter Partes Review No. IPR2015-00440 to -00458

**Declaration of Alan Sherman Regarding
U.S. Patent Nos. 6,963,859; 7,523,072; 8,370,956;
8,393,007; 7,269,576**

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

Dated: December 20, 2014



Alan Sherman

Table of Contents

I. SECTION A: Introduction	1
A. Engagement.....	1
B. Background and Qualifications	2
C. Compensation and Prior Testimony.....	7
D. Information Considered	7
II. Legal Standards for Patentability.....	8
A. Anticipation.....	9
B. Obviousness.....	10
III. The Stefik Patents	16
A. Effective Filing Date and Prosecution History	16
B. The Person of Ordinary Skill in the Art	18
IV. Background: The State of the Art in 1994.....	19
A. Software Copy Protection Techniques and Tamper-Resistant Modules	19
1. <i>Overview</i>	<i>19</i>
2. <i>Tamper-Resistant Modules</i>	<i>20</i>
a. <i>ABYSS</i>	<i>22</i>
b. <i>Superdistribution.....</i>	<i>25</i>
c. <i>FIPS Pub 140-1</i>	<i>27</i>
3. <i>Trusted Systems.....</i>	<i>29</i>
4. <i>Associating Digital Documents and Usage Rights.....</i>	<i>32</i>
5. <i>Using Digital Signatures.....</i>	<i>35</i>
6. <i>File Structure and Protecting “Content”</i>	<i>36</i>
B. Digital Libraries	39
C. Many Copy Protection Schemes Were Being Extensively Designed, Discussed, Implemented, and Refined in 1994.....	41
1. <i>Perritt</i>	<i>45</i>

2.	<i>Kahn</i>	46
3.	<i>Linn</i>	47
4.	<i>Dyad</i>	48
D.	Secure Sessions, Public-Key Encryption, and Certificate Authorities Were Well-Known in the Art	49
V.	SECTION B: General Issues Related to My Patentability Analysis	62
A.	Overview of the Stefik Patent Specification	62
B.	The Claims of the Stefik Patents I Am Addressing in this Report	67
C.	Interpretation of Certain Claim Terms	68
1.	<i>“Usage Rights” and “Manner of Use”</i>	69
2.	<i>“Repository”</i>	81
3.	<i>Distributed repository</i>	88
4.	<i>Recipient computing device / recipient apparatus / sending computing device / the at least one recipient computing device has been determined to be trusted to receive the digital content</i>	91
5.	<i>Digital content / content / digital work</i>	94
6.	<i>Digital document</i>	97
7.	<i>“Identification certificate” / “digital certificate”</i>	99
8.	<i>Rendering system / video system</i>	101
9.	<i>Rendering device / execution device / rendering engine</i>	102
10.	<i>Document Platform</i>	103
11.	<i>Authorization object</i>	105
12.	<i>Registration message / registration identifier</i>	106
13.	<i>’859 patent: means for storing the content</i>	109
14.	<i>’859 patent: means for communication with a master repository</i>	110
15.	<i>’576 patent: means plus function elements</i>	111
D.	Prior Art References	115
1.	<i>Exhibit 1016 – ABYSS</i>	116
2.	<i>Exhibit 1017 – Denning</i>	116
3.	<i>Exhibit 1057 – Dyad</i>	117

4.	<i>Exhibit 1058 – Linn</i>	118
5.	<i>Exhibit 1059 – Kahin</i>	119
6.	<i>Exhibit 1018 – Kahn</i>	120
7.	<i>Exhibit 1020 - Rosen</i>	120
8.	<i>Exhibit 1021 - Hartrick</i>	121
9.	<i>Exhibit 1061 – ABYSS 1990</i>	121
VI. SECTION C: ABYSS (Exhibit 1016)		122
A.	Overview of ABYSS	122
1.	<i>Basic Architecture of ABYSS</i>	125
2.	<i>The ABYSS Supervisor Process</i>	127
3.	<i>Supervisor Keys</i>	129
4.	<i>Creating Protected Software</i>	132
5.	<i>The “Right-to-Execute”</i>	133
6.	<i>Downloading Protected Software</i>	138
7.	<i>Authorization Processes (Installing Software)</i>	139
8.	<i>Executing Software</i>	141
9.	<i>Purchasing, Borrowing, or Transferring Software</i>	146
10.	<i>Software Lending Library</i>	148
B.	Implementing the ABYSS System	150
1.	<i>Integrating Well-Known Public-Key Cryptography Techniques into the ABYSS System, Such as Those Described in Denning</i>	150
a.	Benefits of Using Public Key and Certificate-Based Schemes in ABYSS Deployments.....	154
b.	Developments After 1987 and Before 1994 Improved the Public Key Infrastructure.....	156
c.	Data Integrity and Authentication Functionality	159
d.	Using Public Key Techniques in Authorization Processes	162
2.	<i>The Software Lending Library</i>	163
a.	Distributing Software, Keys, and Certificates for the Library	165

b.	Borrowing a Copy of the Software from the Library...	168
3.	<i>Implementation Variations</i>	172
a.	Usage Rights Language	172
b.	Distributing Other Types of Digital Content.....	174
c.	Deploying ABYSS and Hartrick to Distribute Softcopy Books	177
C.	ABYSS Shows “Usage Rights” and a “Repository”	179
1.	<i>ABYSS Shows “Usage Rights”</i>	<i>179</i>
2.	<i>ABYSS Shows Use of a “Repository”</i>	<i>181</i>
D.	ABYSS Would Have Made the ’859 Patent Obvious.....	187
1.	<i>The Claims of the ’859 Patent</i>	<i>187</i>
a.	Claim 1.....	188
b.	Claims 29 and 58 of the '859 Patent	190
c.	Claims 3, 31, 60 and Claims 5, 33, 62 of the '859 Patent.....	191
d.	Claims 8, 36, 65 and Claims 13, 40, 69 of the '859 Patent.....	193
e.	Claims 15, 42 and 71 of the '859 Patent	193
f.	Claims 16, 43, 72 of the '859 Patent.....	194
g.	Claims 19, 46, 75 and Claims 20, 47, 76 of the '859 Patent	194
h.	Claims 21, 48, 77 of the '859 Patent.....	195
i.	Claims 24, 51, 81 of the '859 Patent.....	196
2.	<i>ABYSS Would Have Made the ’072 Patent Obvious</i>	<i>197</i>
a.	Claim 1 of the '072 Patent	199
b.	Claim 10.....	200
c.	Claim 18.....	202
d.	Claims 8, 16, and 24	204
3.	<i>The Claims of the ’956 and ’007 Patents</i>	<i>205</i>
a.	Claims 1, 7, and 13 (’956) and 1, 6, and 11 (’007)	205
b.	Claims 2, 8, 14 (’956).....	212
c.	Claims 3, 9, 15 (’956) and 2, 7, 14 (’007).....	212

d.	Claims 4, 10, 16 ('956) and 3, 8, 15 ('007).....	213
e.	Claims 5, 11, 17 ('956) and Claims 4, 9, 14 ('007).....	215
f.	Claims 6, 12, 18 ('956) and Claims 5, 10, 15 ('007)....	218
4.	<i>Analysis of the Claims of the '576 Patent</i>	220
a.	Claim 18 of the '576 Patent.....	220
b.	Claim 19 of the '576 Patent.....	224
c.	Claim 21 of the '576 Patent.....	224
d.	Claim 24 of the '576 Patent.....	225
e.	Claim 25 of the '576 Patent.....	226
f.	Claims 26 of the '576 Patent	226
g.	Claim 27 of the '576 Patent.....	226
h.	Claim 29 of the '576 Patent.....	227
i.	Claim 32 of the '576 Patent.....	227
j.	Claim 34 of the '576 Patent.....	228
VII.	SECTION D: Kahn (Exhibit 1018) and Linn (Exhibit 1057).....	229
A.	Overview of Kahn (Ex. 1018)	229
1.	<i>The Repository System</i>	229
2.	<i>Digital Objects and Terms & Conditions of Use</i>	230
3.	<i>Features of the Distributed System</i>	234
4.	<i>Cryptographic Security</i>	237
5.	<i>Registration and Purchase Transactions</i>	242
B.	Overview of Linn (Ex. 1058).....	251
1.	<i>Information Objects</i>	253
2.	<i>Rendering Software</i>	259
C.	Tamper-Proof Hardware.....	262
1.	<i>Overview of Dyad</i>	262
2.	<i>Overview of Shear</i>	264
D.	Kahn and Linn Suggest a Combined Server/Client System	266

E.	Kahn in View of Linn Suggests a System that Will Use “Usage Rights”, “Repositories, and “Authorization Objects”	276
1.	<i>A Kahn / Linn System Would Use “Usage Rights”</i>	<i>276</i>
2.	<i>A Kahn / Linn System Would Use “Repositories”.....</i>	<i>279</i>
a.	The Kahn Repository Servers.....	279
b.	The Linn End-User Devices Are “Repositories”.	284
3.	<i>A Kahn / Linn System Would Use “Authorization Objects” ..</i>	<i>287</i>
F.	A Kahn / Linn System Could Be Configured to Have Additional Security Capabilities Based on the Guidance in Dyad or Shear	289
1.	<i>Dyad.....</i>	<i>289</i>
2.	<i>Shear</i>	<i>290</i>
G.	Kahn in view of Linn Would Render the Stefik Patents Obvious to a Person of Ordinary Skill.....	293
1.	<i>The Independent Claims of the ’859 Patent.....</i>	<i>293</i>
2.	<i>Analysis of the Dependent Claims of the ’859 Patent</i>	<i>298</i>
a.	Claims 3, 31, 60 and Claims 5, 33, 62 of the ’859 Patent	298
b.	Claims 8, 36, 65 and Claims 13, 40, 69 of the ’859 Patent	299
c.	Claims 15, 42, 71 of the ’859 Patent	300
d.	Claims 16, 43, 72 of the ’859 Patent	301
e.	Claims 19, 46, 75 and Claims 20, 47, 76 of the ’859 Patent	302
f.	Claims 21, 48, 77 of the ’859 Patent Would Have Been Obvious Considering Dyad	303
g.	Claims 24, 51, 81 of the '859 Patent.....	303
3.	<i>The Claims of the ’072 Patent</i>	<i>305</i>
a.	Claim 1 of the '072 Patent	306
b.	Claim 10 of the ’072 Patent.....	308
c.	Claim 18 of the '072 Patent	310
d.	Claims 8, 16, and 24 of the '072 Patent	312
4.	<i>The Claims of the ’956 and ’007 Patents</i>	<i>312</i>

a.	Claims 1, 7, and 13 ('956) and 1, 6, and 11 ('007)	313
b.	Claims 2, 8, 14 of the '956 Patent	320
c.	Claims 3, 9, 15 of the '956 Patent and 2, 7, and 14 of the '007 Patent	321
d.	Claims 4, 10, 16 ('956) and 3, 8, 16 ('007)	322
e.	Claims 5, 11, 17 ('956) and Claims 4, 9, 14 ('007)	325
f.	Claims 6, 12, 18 ('956) and Claims 5, 10, 15 ('007)	329
5.	<i>The Claims of the '576 Patent</i>	332
a.	Claim 18 of the '576 Patent	332
b.	Claim 1 of the '576 Patent	337
c.	The Preamble of Claim 1 of the '576 Patent	339
d.	Second Element of '576 Claim 1: “a rendering engine”	340
e.	Third Element of '576 Claim 1: “storage”	341
f.	Fourth Element of '576 Claim 1: “means for requesting use of the digital content stored in the storage”	342
g.	Fifth Element of '576 Claim 1: “a repository coupled to the rendering engine...”	343
h.	Sixth Element of '576 Claim 1: “means for processing a request from the means for requesting.”	343
i.	Seventh Element of '576 Claim 1: “means for checking whether the request is for a permitted rendering...”	347
j.	Eighth Element of '576 Claim 1: “means for processing the request to make the digital content available...”	350
k.	Ninth Element of '576 Claim 1: “means for authorizing the repository for making the digital content available for rendering...”	353
l.	Tenth Element of '576 Claim 1: “means for making a request for an authorization object...”	356
m.	Eleventh Element of '576 Claim 1: “means for receiving the authorization object when it is determined that the request should be granted.”	356
6.	<i>Claims 2 and 19 of the '576 Patent</i>	357

7.	<i>Claims 4 and 21 of the '576 Patent</i>	358
8.	<i>Claims 7 and 24 of the '576 Patent</i>	359
9.	<i>Claims 8 and 25 of '576</i>	359
10.	<i>Claims 9 and 26 of the '576 Patent</i>	360
11.	<i>Claims 10 and 27 of the '576 Patent</i>	361
12.	<i>Claims 15 and 32 of the '576 Patent</i>	362
13.	<i>Claim 29 of the '576 Patent</i>	362
14.	<i>Claim 34 of the '576 Patent</i>	363

VIII. SECTION E: Rosen (Exhibit 1020) in View of Hartrick (Exhibit 1021)

.....	364
A. Overview of Rosen (Ex. 1020)	364
1. <i>Transaction Devices</i>	368
2. <i>Electronic Objects & Tickets</i>	372
3. <i>Certificate Authority</i>	374
4. <i>Communications and Transfers</i>	378
5. <i>Merchants and the Transaction Protocol</i>	384
6. <i>Acquiring Credentials</i>	389
B. Overview of Hartrick	391
1. <i>Specialized SGML Tags</i>	392
2. <i>Royalty Payment Program</i>	400
C. A Person of Ordinary Skill Would Have Integrated Hartrick's SGML Tag Technique in the Rosen System	402
1. <i>Storing SGML tags in the "Usage" field</i>	403
2. <i>Modifying the Trusted Agents</i>	405
3. <i>Tuning the Cryptographic Protocols</i>	408
D. Implementing Rosen to Use the Hartrick Scheme Would Create a System that Uses "Usage Rights" and "Repositories"	408
1. <i>The Rosen / Hartrick System Would Use "Usage Rights"</i>	<i>408</i>
2. <i>A Rosen / Hartrick System Would Use "Repositories"</i>	<i>411</i>

E.	Rosen and Hartrick Would Have Rendered Stefik Patents Obvious.....	414
1.	<i>The Independent Claims of the '859 Patent.....</i>	<i>414</i>
a.	Claim 1 of the '859 Patent.....	415
b.	Claims 29 and 58 of the '859 Patent	418
2.	<i>The Dependent Claims of the '859 Patent.....</i>	<i>419</i>
a.	Claims 3, 31, 60 and Claims 5, 33, 62 of the '859 Patent.....	419
b.	Claims 8, 36, 65 and Claims 13, 40, 69 of the '859 Patent.....	420
c.	Claims 15, 42, 71 of the '859 Patent	421
d.	Claims 16, 43, 72 of the '859 Patent	421
e.	Claims 19, 46, 75 and Claims 20, 47, 76 of the '859 Patent	422
f.	Claims 21, 48, 77 of the '859 Patent	423
g.	Claims 24, 51, 81 of the '859 Patent	424
3.	<i>The Claims of the '072 Patent</i>	<i>425</i>
a.	Claim 1 of the '072 Patent	426
b.	Claim 10 of the '072 Patent.....	429
c.	Claim 18 of the '072 Patent.....	431
d.	Claims 8, 16, and 24 of the '072 Patent	434
4.	<i>The Claims of the '956 and '007 Patents</i>	<i>435</i>
a.	Claims 1, 7, and 13 ('956) and 1, 6, and 11 ('007)	435
b.	Claims 2, 8, 14 ('956).....	443
c.	Claims 3, 9, 15 ('956) & 2, 7, 14 (007).....	444
d.	Claims 4, 10, 16 ('956) and 3, 8, 16 ('007).....	445
e.	Claims 5, 11, 17 ('956) and Claims 4, 9, 14 ('007).....	448
f.	Claims 6, 12, 18 ('956) and Claims 5, 10, 15 ('007)....	452
5.	<i>Analysis of the Claims of the '576 Patent.....</i>	<i>454</i>
a.	Claim 1 of the '576 Patent	454
b.	Claim 18 of the '576 Patent	462
c.	Claims 2 and 19 of the '576 Patent	466

d.	Claims 4 and 21 of the ‘576 Patent	467
e.	Claims 7 and 24 of the ‘576 Patent	467
f.	Claims 8 and 25 of the ‘576 Patent	468
g.	Claims 9 and 26 of the ‘576 Patent	468
h.	Claims 10 and 27 of the ‘576 Patent	468
i.	Claims 15 and 32 of the ‘576 Patent	469
j.	Claim 29 of the ‘576 Patent	469
k.	Claim 34 of the ‘576 Patent	470

IX. Conclusion.....	470
----------------------------	------------

I. SECTION A: Introduction

A. Engagement

A1. I have been retained by counsel for Apple Inc. as an expert witness in the above-captioned proceeding. I have been asked to provide my opinion about the state of the art of the technology described in six patents that issued to Mark Stefik and Peter Pirolli (the “Stefik patents”). Those patents are: U.S. Patent Nos. 7,523,072 (“the ’072 patent”), 6,963,859 (“the ’859 patent”), 8,393,007 (“the ’007 patent”), 8,370,956 (“the ’956 patent”), 7,269,576 (“the ’576 patent”), and 7,225,160 (“the ’160 patent”). These patents all depend, through various series of applications, to a common parent application, and they share a similar, though non-identical, written description.

A2. I also have been asked to provide my opinion on the patentability of certain claims from each of the Stefik patents. Those claims are:

<u>Patent</u>	<u>Claims</u>
’859 Patent	1, 3, 8, 13, 15, 16, 19, 20, 21, 24, 29, 31, 33, 36, 40, 42, 43, 46-48, 51, 58, 60, 62, 65, 69, 71, 72, 75, 76, 81
’072 Patent	1, 8, 10, 16, 18, 24
’956 Patent	1-18
’007 Patent	1-15
’576 Patent	1, 2, 4, 7, 8, 9, 10, 15, 18, 19, 21, 24, 25, 26, 27, 29, 32, 34
’160 Patent	1, 2, 3, 6, 9 and 10

A3. The following is my written report on these topics as they relate to the ’859, ’072, ’956, ’007, and ’576 patents.

B. Background and Qualifications

A4. My Curriculum Vitae is submitted herewith as Exhibit 1014.

A5. I am an expert in the field of cryptography¹ and information security.

Throughout this declaration, I will refer to the field of cryptography and information security as the relevant field or the relevant art. In formulating my opinions, I have relied upon my training, knowledge, and experience in the relevant art. My current *curriculum vitae* (Exhibit 1014) provides a comprehensive description of my academic and employment history, my publications for the previous ten years, and my expert testimony during the previous four years.

A6. As an expert in the field of cryptology² and information security since prior to 1994, I am qualified to provide an opinion as to what a person of ordinary skill in the art would have understood, known, or concluded as of 1994.

A7. I have been working as a researcher, college professor, and private consultant in the area of computer and communications security for over twenty-

¹ Cryptography is the practice of securing communications in the presence of third parties, including through the use of codes and ciphers.

² Cryptology comprises cryptography and cryptanalysis, where cryptanalysis is the practice of breaking codes and ciphers.

five years. My areas of expertise include security of voting systems, information assurance, cryptology, and discrete algorithms.

A8. Currently, I am professor of computer science (with tenure) in the Department of Computer Science and Electrical Engineering (CSEE) at the University of Maryland, Baltimore County (UMBC). At UMBC, I am also director of the UMBC Center for Information Security and Assurance (CISA), which is recognized by the Department of Defense (DoD) and the Department of Homeland Security (DHS) as one of the national centers of academic excellence in information assurance/cybersecurity education and research (CAE and CAEr). At UMBC I teach courses in electronic voting systems, information assurance, cryptology, algorithms, and discrete mathematics. I have mentored six PhD students and twenty-three MS students through completion of their degrees.

A9. As part of my education and experience in computer and communications security I have worked extensively with cryptographic protocols, encrypted sessions, key distribution, encryption, authentication, hash functions, digital signatures, and message authentication codes, which are widely-used building blocks of secure systems. For example, I have designed and analyzed protocols to negotiate a cryptographic context and to establish keys for encrypted sessions. I have also used hash functions, digital signatures, and message

authentication codes in the design and analysis of cryptographic protocols and cryptographic voting systems.

A10. I earned my ScB in mathematics, *magna cum laude*, from Brown University in 1978, and my SM in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT) in 1981. I earned my PhD in computer science from MIT in 1987, studying under Professor Ronald Rivest. Rivest is the co-inventor of the RSA cryptosystem and digital signature system.

A11. I have published numerous papers in refereed archival research journals and conferences. This research includes works on the analysis of cryptographic protocols and cryptographic functions, design of the Scantegrity voting system, analysis of vote verification technologies, algorithms for key management in large dynamic groups, cryptanalysis, applied cryptology, network security, digital forensics, and theoretical foundations of cryptography. At UMBC, my research has been funded by the National Science Foundation (NSF), DoD, and IBM. I am a co-inventor on four cryptographic USA patents.

A12. As a private security consultant, I have performed security reviews of products and systems including for ID2P Technologies, 2factor, LuxSat, InfoScape, Phoenix Technologies, Trusted Information Systems (TIS), and Mattel. For ID2P I analyzed the security of their SafeIDKey authentication device used in the mortgage sector. At TIS my work included analyzing registration protocols

and other protocols for their RecoverKey product. I have also worked as a cryptologic consultant for NAI Labs at Network Associates, performing security research funded by DoD and the Defense Advanced Research Project Agency (DARPA). This latter research includes design of the Dynamic Cryptographic Context Management (DCCM) and Adaptive Cryptographically Synchronized Authentication (ACSA) security systems. This consulting experience includes designing and analyzing protocols that use encryption, hashing, digital signatures, and message authentication codes.

A13. My work for LuxSat involved analyzing the security of their multimedia broadcast system.

A14. My work on the DCCM project included designing and analyzing new cryptographic protocols for negotiating a cryptographic context and establishing keys for encrypted sessions and for controlling access to broadcast streams and services in large dynamic groups. My work on the ACSA project included designing and analyzing a new authentication method for large streams (including media streams) that used message authentication codes and adaptively chose an appropriate level of authentication strength and speed.

A15. My research on the Scantegrity voting system has been published in top venues for my field, including the IEEE Transactions on Information, Forensics, and Security, the Proceedings of the USENIX Security Conference, and

the Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop (EVT). In 2009 and 2011, the city of Takoma Park, Maryland, used the Scantegrity voting system to elect its mayor. It was the first time any end-to-end voter verifiable voting system with ballot privacy was ever used in any binding municipal election. Protocols within the Scantegrity system use encryption, hashing, and digital signatures.

A16. In 2005-2006, I was a member of the UMBC research group that evaluated vote verification technologies in the context of Maryland elections for the Maryland State Board of Elections (SBE). My primary role was to evaluate security and privacy. I presented our findings at the 2006 USENIX/Accurate Electronic Voting Technology Workshop (EVT06). In summer 2005, I testified as an expert witness for the State of Maryland in *Fox vs. Maryland Department of Budget and Management*, which dealt with access to documents about the Diebold AccuVoteTS (AVTS) voting system. In 2007, I was the organizer and principal investigator of the new NSF-funded VoComp University Voting Systems Competition.

A17. At UMBC, I have engaged in research projects in the design and analysis of new voting systems for Internet voting, digital forensics for cloud forensics, and location authentication. I am especially interested in voting systems based on end-to-end voter-verifiable cryptographic audit logs, such as Scantegrity.

My current projects include designing and analyzing a new voting system called Random Sample Elections. Protocols, key management, hash functions, and digital signatures play an important role in my work on voting systems.

A18. At UMBC, I have mentored six students through successful completion of their PhDs degrees in information security, and I have mentored 23 students through successful completion of their Masters degrees in information security.

A19. As an expert witness for Wowza I have analyzed protocols for streaming media servers, which control access to content. As an expert witness for Apple, I have analyzed systems for electronic rights and transaction management.

A20. In addition to the publications mentioned above, my CV has a complete list of my publications, books, and conference proceedings.

C. Compensation and Prior Testimony

A21. I have been retained as an expert witness on behalf of Apple Inc. (“Apple”) for the above-captioned *Inter Partes* Review (IPR). I am being compensated for my time in connection with this IPR at my standard consulting rate, which is \$500 per hour. My compensation is not affected by the outcome of this matter.

D. Information Considered

A22. The exhibits I considered are listed in Appendix A. My opinions also are based on my education, my years of experience working and performing research in the field of cryptography, and my own recollection of the state of the art in 1994.

II. Legal Standards for Patentability

A23. In expressing my opinions and considering the subject matter of the claims of the Stefik patents, I am relying upon certain basic legal principles that counsel has explained to me.

A24. First, I understand that for an invention claimed in a patent to be found patentable, it must be, among other things, new and not obvious from what was known before the invention was made.

A25. I understand the information that is used to evaluate whether an invention is new and not obvious is generally referred to as “prior art” and generally includes patents and printed publications (*e.g.*, books, journal publications, articles on websites, product manuals, etc.).

A26. I understand that in this proceeding Apple has the burden of proving that the Stefik patents are anticipated by or obvious from the prior art by a preponderance of the evidence. I understand that “a preponderance of the evidence” is evidence sufficient to show that a fact is more likely true than it is not.

A27. I understand that in this proceeding, the claims must be given their broadest reasonable interpretation consistent with the specification. After being construed in this manner, the claims are then to be compared to the information in the prior art.

A28. I understand that in this proceeding, the information that may be evaluated is limited to patents and printed publications. My analysis below compares the claims to patents and printed publications that are prior art to the claims.

A29. I understand that there are two ways in which prior art may render a patent claim unpatentable. First, the prior art can be shown to “anticipate” the claim. Second, the prior art can be shown to have made the claim “obvious” to a person of ordinary skill in the art. My understanding of the two legal standards is set forth below.

A. Anticipation

A30. I understand that the following standards govern the determination of whether a patent claim is “anticipated” by the prior art.

A31. I have applied these standards in my evaluation of whether the claims of the Stefik patents would have been anticipated by the prior art.

A32. I understand that the “prior art” includes patents and printed publications that existed before the earliest filing date (the “effective filing date”)

of the claim in the patent. I also understand that a patent will be prior art if it was filed before the effective filing date of the claimed invention, while a printed publication will be prior art if it was publicly available before that date.

A33. I understand that, for a patent claim to be “anticipated” by the prior art, each and every requirement of the claim must be found, expressly or inherently, in a single prior art reference as recited in the claim. I understand that claim limitations that are not expressly described in a prior art reference may still be there if they are “inherent” to the thing or process being described in the prior art. For example, an indication in a prior art reference that a particular process complies with a published standard would indicate that the process must inherently perform certain steps or use certain data structures that are necessary to comply with the published standard.

A34. I understand that it is acceptable to consider evidence other than the information in a particular prior art document to determine if a feature is necessarily present in or inherently described by that reference.

B. Obviousness

A35. I understand that a claimed invention is not patentable if it would have been obvious to a person of ordinary skill in the field of the invention at the time the invention was made.

A36. I understand that the obviousness standard is defined in the patent statute (35 U.S.C. § 103(a)) as follows:

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

A37. I understand that the following standards govern the determination of whether a claim in a patent is obvious. I have applied these standards in my evaluation of whether the claims of the Stefik patents would have been considered obvious in November of 1994.

A38. I understand that to find a claim in a patent obvious, one must make certain findings regarding the claimed invention and the prior art. Specifically, I understand that the obviousness question requires consideration of four factors (although not necessarily in the following order):

- The scope and content of the prior art;
 - The differences between the prior art and the claims at issue;
 - The knowledge of a person of ordinary skill in the pertinent art;
- and

- Whatever objective factors indicating obviousness or non-obviousness may be present in any particular case.

A39. In addition, I understand that the obviousness inquiry should not be done in hindsight, but must be done using the perspective of a person of ordinary skill in the relevant art as of the effective filing date of the patent claim.

A40. I understand the objective factors indicating obviousness or non-obviousness may include: commercial success of products covered by the patent claims; a long-felt need for the invention; failed attempts by others to make the invention; copying of the invention by others in the field; unexpected results achieved by the invention; praise of the invention by the infringer or others in the field; the taking of licenses under the patent by others; expressions of surprise by experts and those skilled in the art at the making of the invention; and the patentee proceeded contrary to the accepted wisdom of the prior art.

A41. I understand the combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results. I also understand that an example of a solution in one field of endeavor may make that solution obvious in another related field. I also understand that market demands or design considerations may prompt variations of a prior art system or process, either in the same field or a different one, and that these variations will

ordinarily be considered obvious variations of what has been described in the prior art.

A42. I also understand that if a person of ordinary skill can implement a predictable variation, that variation would have been considered obvious. I understand that for similar reasons, 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 that technique to improve the other device would have been obvious unless its actual application yields unexpected results or challenges in implementation.

A43. I understand that the obviousness analysis need not seek out precise teachings directed to the specific subject matter of the challenged claim, but instead can take account of the “ordinary innovation” and experimentation that does no more than yield predictable results, which are inferences and creative steps that a person of ordinary skill in the art would employ.

A44. I understand that sometimes it will be necessary to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art. I understand that all these issues may be considered to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue.

A45. I understand that the obviousness analysis cannot be confined by a formalistic conception of the words “teaching, suggestion, and motivation.” I understand that in 2007, the Supreme Court issued its decision in *KSR Int’l Co. v. Teleflex, Inc.* where the Court rejected the previous requirement of a “teaching, suggestion, or motivation to combine” known elements of prior art for purposes of an obviousness analysis as a precondition for finding obviousness. It is my understanding that *KSR* confirms that any motivation that would have been known to a person of skill in the art, including common sense, or derived from the nature of the problem to be solved, is sufficient to explain why references would have been combined.

A46. I understand that a person of ordinary skill attempting to solve a problem will not be led only to those elements of prior art designed to solve the same problem. I understand that under the *KSR* standard, steps suggested by common sense are important and should be considered. Common sense teaches that familiar items may have obvious uses beyond the particular application being described in a reference, that if something can be done once it is obvious to do it multiple times, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle. As such, the prior art considered can be directed to any need or problem known in the field of endeavor in November 1994 and can provide a reason for combining the elements of the

prior art in the manner claimed. In other words, the prior art does not need to be directed towards solving the same problem that is addressed in the patent. Further, the individual prior art references themselves need not all be directed towards solving the same problem.

A47. I understand that an invention that might be considered an obvious variation or modification of the prior art may be considered non-obvious if one or more prior art references discourages or lead away from the line of inquiry disclosed in the reference(s). A reference does not “teach away” from an invention simply because the reference suggests that another embodiment of the invention is better or preferred. My understanding of the doctrine of teaching away requires a clear indication that the combination should not be attempted (*e.g.*, because it would not work or explicit statements saying the combination should not be made.

A48. I understand that a person of ordinary skill is also a person of ordinary creativity.

A49. I further understand that in many fields, it may be that there is little discussion of obvious techniques or combination, and it often may be the case that market demand, rather than scientific literature or knowledge, will drive design trends. When there is such a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within their technical

grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense. In that instance the fact that a combination was obvious to try might show that it was obvious. The fact that a particular combination of prior art elements was “obvious to try” may indicate that the combination was obvious even if no one attempted the combination. If the combination was obvious to try (regardless of whether it was actually tried) or leads to anticipated success, then it is likely the result of ordinary skill and common sense rather than innovation.

III. The Stefik Patents

A50. In this report, I address the patentability of the Stefik patents: the '859, '072, '956, '007, and '576.

A. Effective Filing Date and Prosecution History

A51. I understand that each of the six Stefik patents claims priority through a series of applications back to U.S. Application No. 08/344,760, filed on November 23, 1994, now abandoned.

A52. For the '859, '072, '956, '007, and '576 patents, I have used November 23, 1994, in my analysis as the earliest effective filing date for the claims.

A53. I reviewed the file histories of each of the '859, '072, '956, '007, and '576 patents, and to the extent I found them relevant to my analysis, I have cited to them in my discussion below.

A54. I understand that the '160 patent was issued from U.S. Application No. 10/015,951. I understand that the '951 application was a continuation of U.S. Application No. 09/778,001, filed on February 7, 2001. I understand the '001 application was a division of U.S. Application No. 08/967,084, filed on November 10, 1997.

A55. I understand that during prosecution of the '084 application, the specification was amended during prosecution. The specification was amended in several places to de-emphasize the importance of “repositories” and “usage rights” to the claimed “invention.”

A56. For example, the original specification explained that “attaching” usage right to a digital work was a “key feature of the present invention” but the specification of the '160 was changed to remove the “present invention” language and changed “attached” to “associated”:

~~A key feature of the present invention is that~~ Usage rights are permanently ~~“attached” to~~ associated with the digital work. Copies made of a digital work will also have the associated usage rights ~~attached~~. Thus, the usage rights and any associated fees assigned by a creator and subsequent distributor will always remain with a digital work.

A57. As another example, the original specification explained that the enforcement elements of “the present invention” are embodied in repositories – in fact, the sole enforcement mechanism disclosed in the original specification is repositories. *See Section B, below.* The specification of the ’160 patent was amended to describe repositories as just a “preferred embodiment” of enforcement:

The enforcement elements of the ~~present invention~~preferred embodiment are ~~embodied in~~ repositories.

A58. The glossary definition for “digital work” was changed:

~~Any encapsulated digital information.~~Digital content with any associated usage rights. Such digital ~~information~~content may represent music, a magazine or book, or a multimedia composition. ~~Usage rights and fees are attached to the digital work.~~

A59. The glossary definition of “usage rights” was changed:

~~A language for defining~~An indication of the manner in~~of use by~~ which a digital work may be used or distributed, as well as any conditions on which manner of use or distribution is premised.

A60. As I explain in my declaration addressing the ’160 patent, I believe earliest effective filing date to which the ’160 patent can be entitled is December 17, 2001.

B. The Person of Ordinary Skill in the Art

A61. It is my opinion that a person having ordinary skill in the relevant art at the time of invention (*i.e.*, in 1994) is a person with either a Bachelor of Science

degree in Electrical Engineering, Computer Engineering, or Computer Science and two (2) years of industry experience with designing and/or building cryptographic hardware and associated software, or a Master of Science degree in Electrical Engineering, Computer Engineering, or Computer Science.

IV. Background: The State of the Art in 1994

A. Software Copy Protection Techniques and Tamper-Resistant Modules

1. Overview

A62. Since at least the 1970s, people in the software field have been working to develop effective mechanisms for preventing the unauthorized copying and use of software programs. *E.g.*, Ex. 1038 (US 4,278,837 to Best).

A63. Many software copy protection mechanisms are described in patents and other publications from the 1980s and 1990s. *See, e.g.*, Ex. 1030 (The Copy-Protection Wars, PC Magazine) (Jan 14, 1986) (describing the many copy protection products available on the market in 1986); Ex. 1031 (US 4,658,093 to Hellman) at 4:46-63.

A64. In some copy protection schemes, each piece of software or content would contain functionality that would check the user's computer system and license files, and the software or content itself would enforce its own terms of use. *See, e.g.*, Ex. 1032 (US 5,940,504 to Griswold); Ex. 1043 (Kahn-Cerf-88: The World of Knowbots) at 34.

A65. In other copy protection schemes, each user's computer system contained software and/or hardware components that would enforce terms and conditions of use, and it was the user's computer that contained the functionality that would enforce the terms of use of any software or content. *E.g.*, Ex. 1016 (ABYSS); Ex. 1027 (Superdistribution: The Concept and Architecture); Ex. 1019 (Linn – IMA).

2. Tamper-Resistant Modules

A66. One of the weaknesses of copy protection methods is that content providers do not have control over end users' computers. Once a file is transferred, it and the content in it is in the end user's possession. Providers need a way of ensuring that, even though a user possesses a file containing the content, the user cannot directly access that content.

A67. One way to prevent a user from accessing content is to encrypt the file containing the content, and to hide the decryption key from the end user. The decryption key can be hidden in the executable code or otherwise obfuscated. While this technique is widely used, even today, it has weaknesses: eventually the decryption key must be loaded into the computer's memory to decrypt the content and enable the user to view or execute it. Because the end user has control over his computer, the decryption key can be discovered by monitoring certain memory and

register locations. Also, obfuscation techniques typically are based on obscurity and therefore risk being detected and exploited by determined adversaries.

A68. One known technique for overcoming this weakness is to incorporate into the user's computer a tamperproof, tamper-resistant, or tamper-responding module, which is a hardware component that can store and process decryption keys and other data but that physically prevents or greatly complicates a user from accessing or viewing its contents. *E.g.*, Ex 1037 (FIPS 140-1) at 3 (PDF); Ex. 1057 (Dyad) at 122.

A69. By the early 1980s, it was widely recognized that tamper-resistant modules were one of the best ways to provide physical security in personal computer systems (*i.e.*, computers that an end user physically possesses, as opposed to a mainframe or server). *See, e.g.*, Ex 1017 (Denning) at 161. A tamper-resistant module can be used to store encryption keys and other sensitive data, which prevents an end user from accessing encrypted information. *See, e.g.*, Ex 1037 (FIPS 140-1) at 3 (PDF); Ex. 1057 (Dyad) at 122.

A70. In copy protection schemes, using a tamper-resistant module to hide content or decryption keys from an end user can make it significantly more difficult for end users to subvert the copy protection mechanism. *See, e.g.*, Ex. 1016 (ABYSS) at 38-40; Ex. 1057 (Dyad) at 122. This benefit of tamper-resistant

modules was well-known by 1994. *See, e.g.*, Ex. 1016 (ABYSS); Ex. 1027 (Superdistribution: The Concept and Architecture); Ex. 1057 (Dyad) at 122.

a. ABYSS

A71. In the late 1980s, IBM developed the ABYSS system, which was a computer architecture where a tamper-resistant processor was incorporated into standard computer systems. Steve White and Liam Comerford describe a copy protection technique that relies on the tamper-resistant module in the end user's computer system as a trusted component that enforces terms and condition of use on software. They described this technique in "ABYSS: A Trusted Architecture for Software Protection," published in 1987 in the Proceedings of the IEEE Computer Society Conference on Security and Privacy. Various aspects of the ABYSS system were described in several publications and multiple patents. *See, e.g.*, Ex. 1016 (ABYSS); Ex. 1034 (ABYSS – Cina); Ex. 1035 (U.S. 5,109,413 to Comerford); Ex. 1028 (EP 139) to Comerford. While these publications each discuss details of the ABYSS system, each of them describes different features of the system or different ways it can be implemented.

A72. I analyze the copy protection system described in ABYSS in detail below. *See Section C, below.* In brief, in the ABYSS system, an end user must install a trusted "protected processor" into his computer system. Ex. 1016 (ABYSS) at 39. The protected processor is a tamper-resistant module that

provides cryptographic security, and ensures that encryption keys remain hidden from a user. The protected processor is covered by a special coating that will erase the memory of the unit if it detects any physical tampering with the processor, for example to view its contents. Ex. 1016 (ABYSS) at 39; *see also* Ex. 1036 (ABYSS – Weingart); Ex. 1057 (Dyad) at 126. The ABYSS system provides copy protection by permitting software to run only if a “Right-to-Execute” that software is stored in the protected processor. Ex. 1016 (ABYSS) at 39-40. Software is distributed in encrypted form, and the Right-To-Execute will contain a decryption key for the software, as well as terms and conditions of use that are specified by the software vendor. *Id.* When a user tries to execute a piece of software, the protected processor will check for a corresponding Right-To-Execute. *Id.* If the Right-To-Execute exists, and the user’s request is within the terms of use specified in the Right-To-Execute, the protected processor will permit execution of the program. *Id.* at 39-40, 44-45. The protected processor will decrypt and execute the encrypted portions of the software, keeping the key and cleartext content hidden from the user. *Id.*

A73. Other articles and patents describe different features of the ABYSS system or different ways it can be implemented. For example, the ABYSS article describes a single processor implementation, while several patent applications describe a co-processor configuration. *E.g.*, Ex. 1035 (’413 to Comerford); Ex.

1028 (EP 139) at 1:37-2:25. In contrast to these other publications, the ABYSS article is written for those in the field, and it explains how various aspects of the system can be configured to fit different implementation and design needs (which I describe in Section C, below).

A74. Several other copy protection schemes are built on top of the ABYSS architecture. For example, in the Dyad system, which was designed by J. D. Tygar and Bennett Yee, an ABYSS processor is added as a coprocessor into standard computer systems. *See Dyad: A System for Using Physically Secure Coprocessors*, J.D. Tygar and Bennet Lee, Ex. 1057 (Dyad). As explained in Dyad, the system leverages the ABYSS protected processors:

Several real instances of physically secure processing exist. . . .

The μ ABYSS [62] and Citadel [64] systems provide physical security by employing board-level protection. The systems include an off-the-shelf microprocessor and some non-volatile (battery backed) memory, as well as special sensing circuitry which detects intrusion into a protective casing around the circuit board. The security circuitry erases the nonvolatile memory before attackers can penetrate far enough to disable the sensors or to read the memory contents from the memory chips. The Citadel system expands on μ ABYSS, incorporating substantially greater processing power; the physical security mechanisms remain identical.

Ex. 1057 (Dyad) at 126

A75. Dyad explains how the secure coprocessor can be used to create secure systems and copy protection schemes, including by implementing mechanisms to (1) check the integrity of the host computer using cryptographic checksums, (2) use digital certificates to certify copies of software, (3) use digital certificates to provide copy protection that prevents a purchased and installed copy of a software product from being copied onto and used on a different computer, and (4) create “tokens” that can be used to transfer the right to execute a program. Ex. 1057 (Dyad) at 127-132.

b. Superdistribution

A76. Another copy protection scheme that relies on a tamper-resistant module in the user’s system to enforce the terms and conditions of use is the Superdistribution system developed by Ryoichi Mori and Masaji Kawahara.

A77. In the Superdistribution system, software is freely distributed along with “terms and conditions of its use and the schedule of fees.” The software can be executed only on computer systems equipped with an S-box, which is “a tamper-resistant electronic device[]” that ensures the “enforcement of the conditions set by the vendors.” Ex. 1027 (Superdistribution: The Concept and Architecture) at 1133.

A78. In the Superdistribution system, software is freely distributed, but it is distributed in encrypted form with terms of use and pricing information attached to

it. When software is executed on a system, the S-box will decrypt the software and will track how much and in what manner it is used. The user will be billed according to the terms and conditions attached to the software on a periodic basis.

Superdistribution is an approach to distributing software in which software is made available freely and without restriction but is protected from modifications and modes of usage not authorized by its vendor. . . . Superdistribution software is distributed over public channels in encrypted form. In order to participate in superdistribution, a computer must be equipped with an S-box ~a digitally protected module containing microprocessors, RAM, ROM, and a real-time clock. The S-box preserves secret information such as a deciphering key and manages the proprietary aspects of the superdistribution system. **A Software Usage Monitor insures the integrity of the system and keeps track of accounting information.** The S-box can be realized as a digitally protected module in the form of a three-dimensional integrated circuit.

Ex. 1027 (Superdistribution: The Concept and Architecture) at 1133 (emphasis added).

A79. Mori and Kawahara explain that the “previous work most similar to superdistribution is the ABYSS architecture developed by White, Comerford, and Weingart at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York.” *Id.* at 1134. They further explain that the ABYSS architecture is similar their system. “The chief difference between superdistribution and the

ABYSS scheme is that superdistribution does not require the physical distribution of tokens or anything else to users of a software product. In other words, ABYSS requires that software be paid for in advance while superdistribution does not.” *Id.* at 1134. While Mori and Kawahara state that ABYSS requires software to be paid for in advance, that is inaccurate because the ABYSS system supported usage pricing for software. Ex. 1034 (ABYSS – Cina) at 21 (“Applications could keep track of when the software is executed. This is done by keeping a counter in the application data store. The application could also keep track of what important instructions were being used within the application. The end user could then be billed for the usage of the application or important functions within the application.”).

c. FIPS Pub 140-1

A80. By 1994, the National Institute of Standards and Technology (NIST) also recognized the benefits of tamperproof modules in providing security. For example, FIPS publication 140-1 (FIPS 140-1) sets forth a standard for cryptographic modules used to protect sensitive but unclassified information. FIPS, which stands for Federal Information Processing Standards, publications are documents that set forth computing standards (*e.g.*, encryption standards) that are to be adhered to in government applications. Ex. 1037.

A81. As the publication of standards for cryptographic modules in 1994 shows, the idea of incorporating a physically secure processors and memory into computers was well-known by 1994.

A82. FIPS 140-1 explains that it “specifies the security requirements that are to be satisfied by a **cryptographic module** utilized within a security system protecting unclassified information within computer and telecommunication systems (including voice systems).” Ex 1037 (FIPS 140-1) at 3 (PDF) (Jan. 1994). FIPS 140-1 explains that it specifies security requirements for the design and implementation of cryptographic modules including: “basic design and documentation . . . , **physical security, software security, operating system security, key management**, [and] cryptographic algorithms” Ex 1037 (FIPS 140-1) at 3 (PDF) (emphasis added).

A83. Protection mechanisms for providing cryptographic security can range across a spectrum, from purely software based mechanisms to completely enveloped tamperproof modules. FIPS 140-1 specifies four levels of security, with each level specifying additional protections that must be incorporated into the system. As FIPS 140-1 explains:

Depending on the security level of a cryptographic module, the physical security mechanisms may be designed such that unauthorized attempts at access, use or modification will either have a high probability of being detected subsequent to the attempt by leaving

visible signs (i.e., tamper evident), or have a high probability of being detected during the attempt (i.e., tamper detection) so that appropriate actions can be taken by the module to protect itself (i.e., tamper response). Generally speaking, Security Level 1 simply requires minimal physical protection through the use of production-grade enclosures, Security Level 2 requires the addition of tamper evident counter measures, Security Level 3 requires the use of strong enclosures with tamper detection and response counter measures for covers and doors, and Security Level 4 requires the use of strong enclosures with tamper detection and response counter measures for the entire enclosure.

Ex 1037 (FIPS 140-1) at 30 (PDF).

A84. A person of ordinary skill in the art would have recognized cryptographic modules could readily be incorporated into existing systems, and that such modules provide benefits in terms of “physical security, software security, operating system security, [and] key management,” amongst other benefits.

3. Trusted Systems

A85. The concept of a “trusted system” was well-known in 1994. Where a user’s system could be relied up to enforce the terms of use in a copy protection scheme, that system could be described as “trusted.” *See, e.g.*, Ex. 1016 (ABYSS) at 48.

A86. For example, the entire ABYSS system is designed to provide “A **Trusted** Architecture for Software Protection.” In the ABYSS system, software vendors can restrict use of their software to “trusted” hardware manufactures. “ABYSS allows each software vendor to distribute only to protected processors made by manufacturers that they trust. This is done by creating Rights-To-Execute which can be loaded only onto processors which have the trusted manufacturer’s supervisor key.” Ex. 1016 (ABYSS) at 48.

A87. Similarly, Dyad describes a mechanism for verifying the authenticity of both the host computer system and any content distributed to it. Ex. 1057 (Dyad) at 127-128, 131-132, 139-140. Dyad explains that a secure coprocessor can be used to “solve the host integrity problem” and ensure a user’s computer system has not been tampered with and can be trusted. Ex. 1057 (Dyad) at 128. It also explains that “[w]hen secure coprocessors are added to a system, however, we can quite easily protect executables from being copied and illegally utilized by attackers.” Ex. 1057 (Dyad) at 131.

A88. Another example of a trusted system architecture is the “reference monitor.” A reference monitor is a centralized component of a computer system through which all file access requests are routed. Ex. 1026 (Moffett) at 4. The reference monitor enforces access rules, and ensures users of the system cannot

circumvent the system's security. For example, one implementation of such a system describes the reference monitor as follows:

We use the Reference Monitor concept . . . to model an access control system. The function of a reference monitor is to enforce the authorized access relationships between subjects (users) and objects of a system. It is a trusted component of the system. . . . [I]n general, access control is carried out by the system, and not by the target object itself . . . The operation of the Reference Monitor is transparent to the user provided the access is authorised, but if the access is not authorised the operation is not invoked and the user is informed by a failure message.

Ex. 1026 (Moffett) at 4.

A89. Another example of a trusted system architecture is described in U.S. Patent No. 5,557,518 to Rosen ("Rosen"), filed on April 28, 1994. Ex. 1020 (Rosen). I analyze the systems described in Rosen in detail below. *See Section E, below.*

A90. Rosen describes a method for distributing media (*e.g.*, software, movies, and other electronic content) in encrypted form along with an associated "ticket" that can be used to decrypt and use the media. Ex. 1020 (Rosen) at 5:59-6:13, 4:54-5:3. A decryption ticket is associated with a particular object through an object identifier, contains a decryption key for the content, and contains usage rights that specify "restrictions on usage of the electronic object." *Id.* at 5:59-6:13,

Fig. 2. In the Rosen system, both merchants and users have “trusted systems” that employ tamper-resistant modules to ensure security of transactions and content.

Id. at 1:5-11 (“[T]he system utilizes tamper-proof electronic units, referred to as ‘trusted agents’ in combination with money modules to create a secure transaction environment for both the buyer and seller of electronic merchandise and services.”).

4. Associating Digital Documents and Usage Rights

A91. A common technique used in systems for distributing and controlling use of digital content in 1994 was to associate a digital document with data or information (“rights”) that defined how and under what conditions that document may be used.

A92. For example, Fischer describes a system that restricts and monitors how computer programs may be used. Ex. 1029 (Fischer) at 2:16-49. As Fischer explains: “For example, the present invention advantageously serves to **bind limitations** to programs” Ex. 1029 (Fischer) at 3:22-25. In the Fischer system, these limitations are called Program Authorization Information (PAI). Ex. 1029 (Fischer) at abstract: “The set of authorities and/or restrictions as signed to a program to be executed are referred to as ‘program authorization information’ (or ‘PAI’). Once defined, the program authorization information is thereafter associated with at least one program to be executed to thereby delineate the

resources and functions that the program is allowed to utilize and/or is not allowed to utilize.”

A93. Another example is shown in Linn (Ex. 1019). Linn describes a technical mechanism for associating a digital object with attributes of this object. Ex. 1019 (Linn - IMA) at 15 (fig 2). Specifically, Linn describes how a digital object can be sealed in a digital “envelope” with attributes of the object visible on the outside of the envelope. In Figure 2 (below), these attributes include rights, for example the number of copies the user is authorized to make. A digital signature cryptographically associates these rights with the object. “Digital signatures can be employed as a tool to ‘seal an envelope’ and verify the authenticity of copyrighted materials distributed over the Network.” Ex. 1019 (Linn - IMA) at 18.

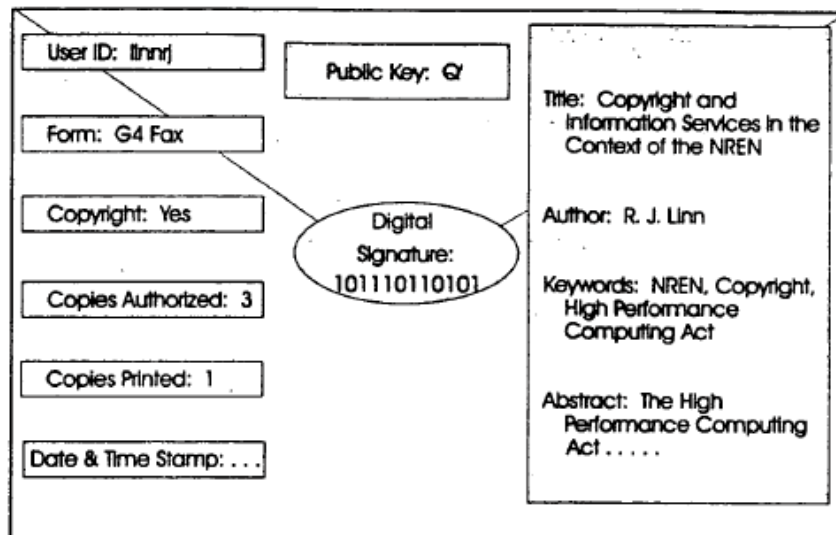
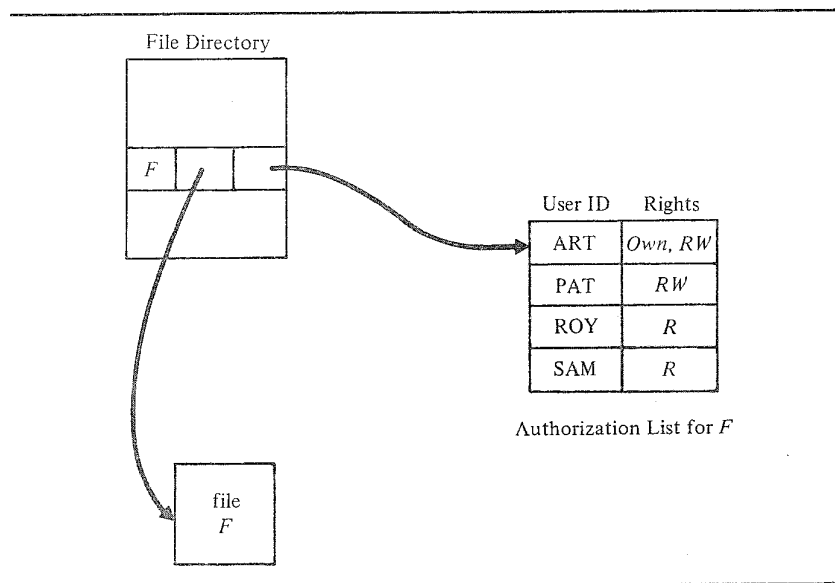


Figure 2: An Information Object -- Envelope plus Content

A94. Many other publications discuss systems that work by associating content with usage rights or terms and conditions of use. *E.g.*, Ex. 1016 (ABYSS); Ex. 1027 (Superdistribution) at 1123.

A95. Denning shows how usage rights would be associated with content in a conventional file system. Ex. 1017 (Denning) at 210. A file directory would contain a file name, a pointer to the content file, and a pointer to an “authorization list” which contains a list of rights associated with the file.

FIGURE 4.12 Authorization list for file.



A96. Fischer shows a similar data structure being used to associate a program with its authorization information. Ex. 1029 at Fig. 3A.

A97. It was also well known how to apply standard cryptography techniques to bind digital information. Examples of these techniques include hash functions, message authentication codes, digital signatures, and encryption. For

example, a digital certificate cryptographically binds a name with some other information (*e.g.*, a public key). This is typically done by digitally signing a list of data that include a name and other information (*e.g.*, a public key). Ex. 1017 (Denning) at 170 (“Upon registering a public transformation E_A with the system, a user A receives a signed certificate from S containing E_A .”), 178-179 (“ S sends to A the certificates $C_A = D_S(A, E_A, T_1) \dots$ ”).

5. Using Digital Signatures

A98. In 1994 it was well known that the authenticity of software and other data could be protected by digital signatures. For example, Denning states, “Merkle [Merk80] describes two applications of signatures for software protection. The first involves distributing network software to the individual nodes of a network. If the software is signed, the nodes can check the validity of the software before execution. The second involves running privileged programs in operating systems.” Ex. 1017 (Denning) at 15.

A99. In 1989, Davida, Desmedt, and Matt described how digital signatures could be used to authenticate the source and integrity of software. Ex. 1033 (Davida) at 313-17. The first two steps of their method show that they are using digital signatures to verify the software. As they state: “The scheme consists of performing the following: 1. The vendor generates signed software. 2. The user is able to verify the signed software. ...” Ex. 1033 (Davida) at 313.

A100. As Denning also explains, it was well known, when using digital signatures, how to use digital certificates to obtain the authenticated public key needed to verify the signature. Ex. 1017 (Denning) at 169-70; *see generally id.* at 164-79 (describing key management and distribution techniques).

6. File Structure and Protecting “Content”

A101. Digital content refers to digital representations of works, such as books, photos, videos, programs, data files. It was well known that methods for protecting one type of digital content will typically work for all types of digital content. For example, Linn states, “All mechanisms are applicable to any information distributed over a computer network” Ex. 1019 (Linn - IMA) at 17.

A102. In general, copy protection techniques that can be applied to one type of digital content can be applied to other types of digital content. This is because all digital content can be, and typically is, represented as a sequences of ones and zeros, and at some level, every digital content protection system protects sequences of ones and zeros.

A103. For example, any computer program can be represented as a sequence of ones and zeros when the executable code is stored in the computer’s volatile or persistent memory (*e.g.*, as an executable file on persistent storage). Ex. 1039 (Aspray). Programs are simply data files. In this sense, a technique for protecting

a computer program will use essentially the same operations that would be used to protect any other kind of “digital content” (*i.e.*, a data file).

A104. I also note that digital documents people consider to be “static” (*e.g.*, a postscript file of an essay, a digital file representing an image, an MP3 file) often will contain executable code that is run when the document is rendered. Ex. 1040 (Thinking in Postscript). Again, at the operational level of zeroes and ones, there are essentially no differences between protecting “programs” and protecting other kinds of digital documents.

A105. A digital content protection system may be viewed as an “interpreter” – a system that “interprets” data (*i.e.*, the digital content) under the control of instructions (including the usage rights). Actions performed by such a system might include, for example, displaying, printing, copying or transferring the content. In this view, there is no essential difference between the various types of digital content (*e.g.*, program vs. book). Ex. 1040 (Thinking in Postscript) at 1-6 (“PostScript as a programming language”).

A106. By 1994, it was well understood how to structure and arrange digital data in blocks and segments in a computer memory. For example, in 1975 Saltzer and Schroeder published a widely-read paper explaining how to protect data from unauthorized use or modification. Ex. 1051 (Saltzer).

A107. Strictly speaking, a “segment” is a logical unit, while a “block” is a syntactic unit that is useful for organizing large segments (*e.g.*, a segment might comprise a sequence of blocks). (I note that the Stefik patents use the word “block” in the sense of “segment” – *see* Ex. 1001 (859) at Figs. 7-9).

A108. It was well understood that data can be arranged into logically separate units called “segments.” Access to a segment can be controlled by controlling a unique identifier that identifies where in computer memory the segment begins. Saltzer explains:

Conceptually, we may achieve this separation by enlarging the function of the memory system to provide uniquely identified (and thus distinctly addressed) storage areas, commonly known as segments. For each segment there must be a distinct addressing descriptor, and we will consider the set of addressing descriptors to be part of the memory system, as in Fig. 5. Every collection of data items worthy of a distinct name, distinct scope of existence, or distinct protection would be placed in a different segment, and the memory system itself would be addressed with two component addresses: a unique segment identifier (to be used as a key by the memory system to look up the appropriate descriptor) and an offset address that indicates which part of the segment is to be read or written.

Ex. 1051 (Saltzer) at 1290.

A109. It was well known that segments can be arranged in a hierarchy known as a “tree.” Saltzer explains: “Then, as in Fig. 13, all of the access

controllers of the system will be arranged in a hierarchy, or tree structure, branching from the first access controller in the system,” Ex. 1051 (Saltzer) at 1299.

B. Digital Libraries

A110. In the late 1980s, scientists and engineers began working to design “digital libraries” that would enable end users to use the Internet to access articles and other content stored in remote locations. *See generally* Ex. 1042 (Samuelson, Digital Library); Ex. 1043 (Kahn-Cerf Digital Library Project).

A111. In a 1988 report written for the Corporation for National Research Initiatives (CNRI), Robert Kahn and Vinton Cerf described a distributed Digital Library System that would enable consumers to use the Internet to search and access content from libraries of books, articles, and other written materials. *See* Ex. 1043 (Kahn-Cerf The Digital Library Project) at 3, 5, 12. Kahn and Cerf recognized in their analysis that the project would face challenges due to the ease with which digital information could be replicated. Ex. 1043 (Kahn-Cerf) at 11.

A112. In their report, Kahn and Cerf proposed a technological solution to preventing the unauthorized replication of digital works that they termed “Knowbots.” Ex. 1043 (Kahn-Cerf-88) at 34. Knowbots are software applications that intermediate requests from users to access content. For example, Kahn and

Cerf describe a special trusted knowbot that is attached to a digital work and enforces usage rights if the digital work is transferred between two systems.

A class of trusted Knowbots called couriers have the special responsibility to look after selected objects on behalf of their authors or other owners of rights in the objects. A courier may be entrusted with responsibility for an entire database or a specific document or only a portion of it. . . . For purposes of this discussion, we assume that all documents are entrusted to couriers and never appear in the library system without an accompanying courier to protect the owner's rights. The combination of a courier and its accompanying entity (e.g., paragraph, document, database) is controlled object in the system.

When a controlled object is provided to a user, all access to its contained entity is handled via its courier. If the owner of the entity originally wished to charge on a per use basis, the courier will be instructed to report such usage when it actually occurs, or to seek permission for use immediately beforehand and to deny access if it cannot be granted. Should a user wish to extract a portion of the controlled object, say for inclusion in another document, a new courier and controlled object would be created to convey the information and to represent the owner's potential interest in the user's new work.

Ex. 1043 (Kahn-Cerf-88) at 34 (emphasis added).

A113. The CNRI also held a workshop about Digital Libraries on May 18 and 19, 1989, entitled “Workshop on the Protection of Intellectual Property Rights

in a Digital Library System: Knowbots in the Real World.” The participants discussed and analyzed approaches to protecting digital content made available through a digital library.

A114. Mark Stefik has recognized that Kahn’s and Cerf’s article about digital libraries was influential. In 1996, Stefik wrote a forward to his 1994 article, “Letting Loose the Light,” where he acknowledged that the industry, and Kahn and Cerf in particular, had been considering copy protection schemes since the late 1980’s.

In “The Digital Library Project: The World of Knowbots” in Part I, Robert Kahn and Vinton Cerf ask, “If a thousand books are combined on a single CD-ROM and the acquirer of the CD-ROM only intends to read one of them, what sort of royalty arrangement is appropriate to compensate the copyright owners? How would compensation be extended for cases in which electronic copies are provided to users?”

Their questions show how, in 1988, issues about copyright protection and payment for using information arose in the context of early CD-ROM distribution.

Ex. 1044 (Internet Dreams - Stefik foreword) at 222 (emphasis added).

C. Many Copy Protection Schemes Were Being Extensively Designed, Discussed, Implemented, and Refined in 1994

A115. In the early 1990s, it was widely recognized that the Internet would be a valuable medium for distributing software and other digital content. It also was recognized that the digital distribution of content over the Internet would present

challenges in preventing unauthorized copying and use of digital content. *See* Ex. 1019 (Kahin) at 1 (“Owners of content – text, images, music, motion pictures – are understandably fearful of releasing proprietary information into an environment which is lacking in security and has no accepted means of accounting for use, and copying.”); Ex. 1043 (Kahn-Cerf-88) at 11 (“There are many issues at stake in this area, not the least of which relate to the ease with which information can be replicated once in digital form and the rapidity with which large quantities of information can be processed (accessed, transferred, analyzed, integrated, etc.).”); Ex. 1045 (CNRI Digital Library Workshop) at 9-10.

A116. As evidenced in the literature documenting the conferences, workshops, and consortiums, it was widely recognized that there was a need to develop an effective copy protection mechanism, and many people in the field were working on developing solutions this problem.

A117. In April 1993, the Interactive Multimedia Association (IMA), in conjunction with the Coalition for Networked Information (CNI), the John F. Kennedy School of Government, and the Massachusetts Institute of Technology, held a conference titled Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment. Researchers presented descriptions of various copy protection mechanisms that were under development.

A118. The publications that were presented at the Conference were collected into a Proceedings, which was published in the Journal of the Interactive Multimedia Association Intellectual Property Project in January 1994. In an article by Brian Kahin summarizing the proceedings, entitled “The Strategic Environment for Protecting Multimedia,” Kahin described the conference as follows:

The advent of distributed computing over high-bandwidth wide-area networks looks like a worst-case scenario for intellectual property.

Owners of content – text, images, music, motion pictures – are understandably fearful of releasing proprietary information into an environment which is lacking in security and has no accepted means of accounting for use, and copying. The variety of formats and the variety of proprietary interests involved complicate the problem and attempts at solutions.

On April 2 and 3, 1993, four organizations involved in networking and multimedia issues sponsored a two-day workshop at Harvard's John F. Kennedy School of Government to address the problem. These organizations – the Coalition for Networked Information, the Interactive Multimedia Association, the MIT Program on Digital Open High Resolution Systems, and the Information Infrastructure Project in the Kennedy School's Science, Technology and Public Policy Program – represented **a set of different perspectives on what all saw as a broad common problem.** The workshop was designed to:

- map the territory between secure systems and the need for practical, user-friendly systems for marketing information resources and services;
- **survey the technological landscape, evaluate the potential benefits and risks of different mechanisms**, define a research agenda, and frame related implementation and policy issues;
- **consider how and where within the overall infrastructure different technologies are best implemented**; and
- present and analyze models for explaining protection systems and strategies.

Ex. 1019 (IMA - Kahin) at 1 (emphasis added). As Kahin explains, the topics discussed addressed various aspects of the distribution and protection of content.

A119. The articles in the Proceedings address the technologies that fit into different parts of the overall infrastructure, and consequently, would be combined with other techniques in any implementation of a “digital rights management” system. For example, as Kahin explained:

Speakers were invited to address these issues along with the potential roles of particular technologies and mechanisms: billing servers; type-of-service identifiers; header descriptors; labeling and tagging; fingerprinting; digital signatures; contracting mechanisms; EDI (electronic data interchange); **copy protection**; serial copy management; **authentication servers**; **software envelopes**; encryption; **display-only systems**; **concurrent use limitations**; and **structured charging**.

Ex. 1019 (Kahin) at 1 (emphasis added).

1. Perritt

A120. Henry Perritt presented an article (“Perritt”) at the IMA conference about attaching “permission headers” to digital content to allow content to be stored, searched, and accessed through digital libraries. His article also discussed using Knowbots to access and protect that content, building on the foundation Cerf and Kahn had described 6 years prior. Ex. 1019 (IMA – Perritt) at 29.

The phrase “digital library” and the basic concept were first articulated in a 1989 report growing out of a workshop sponsored by the Corporation for National Research Initiatives. From its inception, the digital library concept envisioned retrieval of complete information resources and not merely bibliographic information.

Ex. 1019 (IMA – Perritt) at 29.

A121. Perritt explained that the permissions header contained fields for specifying usage rights information, such as whether rights could be transferred, and the price for acquiring those rights:

[T]he following kinds of privileges in the requester should be addressed in the permissions header: . . . • use privilege, either unrestricted or subject to restrictions; • copying, either unlimited or subject to restrictions like quantitative limits; • distribution, either-unlimited or subject to restrictions, like geographic ones or limits on the markets to which distribution can occur

In addition to these discrete privileges, the permissions header must express pricing information. The most sensible way of doing this is to have a price associated with each type of privilege.

Ex. 1019 (IMA – Perritt) at 31.

A122. Perritt also explained that technologies such as encryption, digital signatures, and certificate authorities can be integrated into digital libraries to ensure integrity and authenticity. Ex. 1019 (IMA – Perritt) at 32-33 (discussing public key encryption, “hashing” and digital signatures, encryption options, etc.).

2. Kahn

A123. Kahn presented a paper on an Electronic Copyright Management System that discussed digital libraries as secure repositories of content. Kahn explained that he was actively developing a test bed for his system to be used with the Library of Congress. Ex. 1019 (IMA – Kahn) at 112. Kahn envisioned a “set of distributed repositories” to provide wide access to content. Ex. 1019 (IMA – Kahn) at 114. Kahn described the library as including “Rights Management System” that would enable the transfer of rights to view, modify, or otherwise use content. Ex. 1019 (IMA – Kahn) at 115-116.

A124. Kahn’s IMA paper essentially is a summary of the Kahn patent, which I discuss in detail in my patentability analysis below. *See Section D, below.*

A125. In the Kahn system, the author or creator of content can specify usage rights, conditions, and fees, and then deposit the content in a publicly accessible

“repository.” Ex. 1019 (IMA – Kahn) at 113, 116. Anyone can search the publically available repositories, and acquire a digital copy of a work registered and deposited in a repository. Kahn envisioned that a user acquiring a copy of a work could only use the work in accordance with the conditions specified by the author, which the user could acquire through an online transaction. Ex. 1019 (IMA – Kahn) at 115.

A126. Kahn proposed using a certificate authority to provide authentication services in his system. He leveraged the Privacy Enhanced Mail framework described by the IETF to provide those services. Ex. 1019 (IMA – Kahn) at 111 (“‘Digital signatures’ and ‘privacy enhanced mail’ will be used for registration and transfer of exclusive rights and other copyright related documents.”), 112 (the testbed will use “Privacy enhanced mail and digital signatures to facilitate on-line transactions”), 116-117.

3. Linn

A127. In “Copyright and Information Services in the Context of the National Research and Education Network,” R.J. Linn describes a set of mechanisms for protecting electronic content. Ex. 1019 (IMA – Linn) at 14-25.

A128. In the Linn system, content is distributed in an encrypted and signed envelope that includes basic usage rights information on it. Ex. 1019 (IMA – Linn) at 14-15. Linn explains that users will use “rendering software” to process,

decrypt, and display the content in each object. In addition, “[c]opies of objects are exchanged using the rendering software.” Ex. 1019 (IMA – Linn) at 15.

A129.Linn explains that the “rendering software” will enforce usage rights associated with the envelope. Ex. 1019 (IMA – Linn) at 15 (“Two active-mechanisms *implemented in the rendering software* will achieve the requirements for protection outlined in the previous section.” (emphasis in original)).

A130. The content is permanently stored in the envelope, and each time the object is executed or accessed, the rendering software will check the usage rights and validate the digital signature before permitting access to the content of the object. Ex. 1019 (IMA – Linn) at 15-16.

4. Dyad

A131.J.D. Tygar and Bennet Yee describe the Dyad system, which used a secure processor to ensure that a client computer system was trusted and ensure copy protection schemes were enforced. I discussed the Dyad system in ¶¶ A74, A87, *above*. Dyad discloses a copy protection mechanism that builds on the system described in ABYSS. *Id.* In the Dyad system, software is distributed in encrypted form, and decrypted inside of the secure coprocessor, thus protecting both the encryption keys and the cleartext data. Ex. 1057 (Dyad) at 137, 140-141.

A132.Tygar and Yee explain that the use of secure coprocessors in their system can enable a powerful copy protection scheme:

When secure coprocessors are added to a system, however, we can quite easily protect executables from being copied and illegally utilized by attackers. The proprietary code to be protected-or at least some critical portion of it-can be distributed and stored in encrypted form, so copying it without obtaining the code decryption key is futile. Public key cryptography may be used to encrypt the entire software package or a key for use with a private key system such as DES. **When a user pays for the use of a program, a digitally signed certificate of the public key used by his secure coprocessor is sent to the software vendor.** This certificate is signed by a key management center verifying that a given public key corresponds to a secure coprocessor, and is prima facie evidence that the public key is valid. **The corresponding private key is stored only within the NVM of the secure coprocessor; thus, only the secure coprocessor will have full access to the proprietary software.**

Ex. 1057 (Dyad) at 131-132.

A133. The various mechanisms described in the articles presented at the IMA conference are not mutually exclusive. Instead, they each focus on different aspects of creating a system for securely distributing content, and a person of ordinary skill in the art would understand the techniques could easily be combined.

D. Secure Sessions, Public-Key Encryption, and Certificate Authorities Were Well-Known in the Art

A134. Basic concepts of cryptography include conventional and public-key encryption, digital signature, message authentication code, and cryptographic hash function. These basic concepts had been known and used for years before 1994.

A135. Cryptography provides techniques for confidentiality, authentication, and integrity. Consider two communicants, Alice and Bob, who wish to communicate over an insecure channel. Confidentiality means that unintended recipients (*e.g.*, an eavesdropper Eve) cannot read the plaintext messages. Authentication means, for example, that Bob can convince himself that a message actually came from Alice as the message claims. Integrity means that no one can modify the message without detection. As disclosed for example, by Denning (Ex. 1017), all of these fundamental cryptographic concepts were well known in 1994.

A136. A conventional symmetric or symmetric key encryption function E takes a plaintext message x and a secret key k and mixes these two inputs together in a complicated way to produce a ciphertext $y = E_k(x)$. It is assumed that Alice and Bob have exchanged the secret key k in advance with each other, but that Eve does not know k . If the encryption function is secure, Eve cannot figure out what is x or k simply by observing y . Thus, encryption reduces the problem of keeping a message confidential to the, hopefully, simpler problem of keeping the secret key secret. To decipher the ciphertext, Bob computes $x = E_k^{-1}(y)$, where E^{-1} is the decryption function corresponding to E .

A137. A message authentication code MAC enables the recipient to detect modification of the message. A keyed MAC takes a secret key k and a message x and computes a short tag $t = \text{MAC}_k(x)$. When Alice sends a message x to Bob, she sends both x and t , thus providing integrity and some authentication. Changing the message, even by one bit, will change the tag with overwhelming probability. Without knowledge of the secret key k , Eve cannot compute a correct tag. When Bob receives (x, t) , he verifies that the tag is correct by recomputing it using his knowledge of k . Since both Alice and Bob know k , MACs prove that the message came from one of them (but a third party cannot be certain from which one).

A138. In public-key (also called “asymmetric”) cryptography, the encryption and decryption keys are separate, and knowing the (public) encryption key does not enable one to determine the corresponding (secret) decryption key. This technology greatly simplifies key management, and it enables digital signatures.

A139. Using public-key cryptography, for Alice to send a message to Bob, she would encrypt the message using Bob’s public encryption key. Only Bob knows his secret decryption key, which he uses to decipher the message.

A140. In public-key cryptography, key management is simpler than in conventional cryptography because the communicants do not need to exchange secret keys. Each must, however, obtain the authenticated public key of the other.

As I will explain below, certificates provide a convenient way to accomplish this goal.

A141. In the RSA public-key cryptosystem, Alice can digitally sign a message x by using her private signing key d to compute a signature $s = \text{RSA}_d(x)$. Upon receipt of (x, s) , Bob can verify the signature using Alice's public key e by computing $x' = \text{RSA}_e(s)$ and checking that $x = x'$. In this way, Bob can convince himself that the message actually came from Alice. Unlike with MACs, because Bob does not know Alice's private signing key d , Bob cannot forge Alice's digital signature. In this example, the communication (x, s) provides authentication without confidentiality. If confidentiality is also desired, an extra layer of encryption can be added.

A142. When messages are very long, rather than signing the entire message, it is convenient instead to sign a short "fingerprint" of the message. A cryptographic hash function h compresses an arbitrarily long message x into a short fixed-length fingerprint $t = h(x)$. Fingerprints are also called tags or hash values. Changing any bit of the message results in a completely different and unpredictable tag, with overwhelming probability. Moreover, it is infeasible to find collisions -- any two inputs that hash to the same tag. Similarly, given a tag, it is infeasible to find any input that hashes to the given tag.

A143. Public key encryption was first described in the 1970s, and was a well-known technique in 1994. *See* Ex. 1019 (IMA – Kahn) at 116-117; *see* Ex. 1046 (Diffie-Hellman, New Directions in Crypto 1976).

A144. In public-key cryptography, it is important that each communicant be able to obtain the genuine authenticated public-key of the entity with whom he wishes to communicate. In 1994, it was well known how to accomplish this goal using certificates and certificate authorities. For example, Dorothy Denning describes such techniques in her 1983 textbook *Cryptography and Data Security*. Ex. 1017 (Denning) at 175-179.

A145. A certificate authority is a hierarchical system of authoritative servers that distribute public keys and provide a trusted mechanism for verifiably associating a key with a particular entity. Ex. 1047 (X.509) at 17-21. A digital certificate is a document containing the name or identity of the owner, a public key, the identity of the certificate authority, and a time stamp that is digitally signed by the certificate authority (*i.e.*, using their private signature key, the certificate authority digitally signs a cryptographic hash of the document). Given a public key with a certificate for it, a communicant can verify the authenticity of the public key by verifying the signature of the certificate (using the certificate authority's public key).

A146. In 1994, it was well known how two communicants can establish a secure communications session using either public-key or conventional cryptography. Ex. 1017 (Denning) at 178-79; Ex. 1050 (Needham Schroeder) at 1-4. To do so, the communicants can authenticate each other and establish a symmetric session key to encrypt their communications. When using public-key cryptography, certificates are useful in this protocol to enable each communicant to obtain the authenticated public key of the other. Because symmetric encryption is typically much faster than public-key encryption, it is common first to use public-key cryptography to establish a symmetric session key, and then to use this session key with symmetric cryptography to encrypt the main content of the communications. Establishing the session key is simpler to do using public-key cryptography than with conventional cryptography.

A147. Kerberos (Ex. 1049) is an example of a system that existed in 1994 that, among other services, could establish session keys. The initial version was built using conventional cryptography; there is another version that uses public-key cryptography. Ex. 1049 (Kerberos).

A148. Denning describes a standard technique for initiating a secure communication link by exchanging certificates issued by trusted server S .

A public-key cryptosystem can also be used to exchange session keys . . . Following the timestamp protocol . . . , let C_A and C_B be A 's

and B 's certificates . . . , signed by S . A and B can exchange a key K with the following protocol . . . :

Public-key distribution protocol

1. A sends the plaintext message (A, B) to S , requesting certificates for A and B .

2. S sends to A the certificates

$$C_A = D_S(A, E_A, T_1)$$

$$C_B = D_S(B, E_B, T_1) .$$

3. A checks C_A and gets B 's public transformation E_B from C_B . A then generates a random key K , gets the time T , and sends the following to B :

$$(C_A, C_B, X) ,$$

where $X = E_B(D_A(K, T))$, and D_A is A 's private deciphering transformation.

4. B checks C_B , gets A 's public transformation E_A from C_A , and computes

$$E_A(D_B(X)) = (K, T) ,$$

where D_B is B 's private transformation.

For added protection, the protocol can be extended to include a handshake between B and A , following the approach described for the centralized key distribution protocol. Of course, in a public-key system users can send messages directly using each other's public keys (obtained from certificates). The protocol would be used only if

conventional encryption of messages is needed for performance reasons.

Ex. 1017 (Denning) at 178-179. In this routine, the client A (the device initiating the connection) obtains and validates a digital certificate for a server B (the device being connected to). A then sends to B its certificate as well as a signed session key, which it then encrypts with B 's public key. B validates the certificate and then sends a response that confirms the connection.

A149. In this system, S is a trusted authority, and S digitally signs certificates to certify they are valid. It is assumed that S has distributed its public key to all participants in the system. Each certificate C_A is a message digitally signed by S : $D_S(A, E_A, T_I)$. The message contains the owner's identity (A), the owner's public key (E_A), and a timestamp (T_I). As I mentioned above (§§ A145, A141-A142), the process for validating a digital certificate is essentially the same process used to validate any other digitally signed message: one recomputes the hash of the original message (here, A , E_A , and T_I), decrypts the digital signature using the authority's public key, and compares the two values. If they match, the certificate is valid.

A150. For an entity A to initiate a session with entity B , A acquires B 's digital certificate, and then creates a message containing a session key and a timestamp. A digitally signs the message and then encrypts the signed message

with B 's public key. A then sends its certificate and the encrypted message to B . B can accept the request and initiate a session by returning a message encrypted with the session key. Because only B can decrypt the message containing the session key (only B has the corresponding private decryption key), A knows that if it receives a response encrypted with the session key, it came from B .

A151. Denning explains that the process of initiating a secure link can include a handshake where a random number is exchanged and used to validate the session:

For added protection, the protocol can be extended to include a “handshake” between B and A . . . :

4. B picks a random identifier I and sends to A

$X = E_K(I)$.

5. A deciphers X to obtain I , modifies it in some predetermined way to get I' (e.g., $I' = I + 1$), and returns to B

$E_K(I')$.

This confirms receipt of I and the use of K .

See id. at 175.

A152. Use of the handshake during session initiation or during the course of a session allows B to verify that it is communicating with A (who possesses key K) and that it is not simply receiving “replayed” messages. A replayed message is a message that is intercepted by a third party, saved, and the later retransmitted to the

original destination. The third party may not be able to decrypt the message, but the destination can (since it knows key K). A third party might do this for a message instructing a bank to transfer money; by replaying the transfer instruction 20 times, the third party could turn a single request to transfer \$100 into 20 such requests, and transfer \$2,000 instead of \$100.

A153. In secure communications, a standard cryptographic technique was to include a random number or other unique data in each encrypted message. This prevents an attacker from intercepting messages, and then later “replaying” them by sending them to the recipient. One technique for preventing replay attacks is to include a nonce in each encrypted message. A nonce can be a random number. Ex. 1050 (Needham & Schroeder, Using Encryption for Authentication) at 995 (“[B] is unclear that . . . subsequent messages supposedly from A are not being replayed. To guard against this possibility, B generates a nonce identifier for the transaction.”). Or it can be a timestamp. Ex. 1017 (Denning) at 175 (“The timestamp T guards against replays of previous keys”).

A154. In 1988, the ITU issued the first version of its X.509 standard, which defined an industry standard for creating digital certificates and providing certificate authority services. *See* Ex. 1047 (X.509). Later versions of the X.509 standard are widely used today.

A155. The X.509 standard specifies a particular format for certificates. Ex. 1047 (X.509) at 17-18; Ex. 1041 (Applied Cryptography) at 169-72; Ex. 1048 (RFC 1422) at 6-7.

A156. In 1993, the IETF released a suite of four RFCs for providing Privacy Enhancement for Internet Electronic Mail (PEM), RFC 1421 to RFC 1424. These RFCs were the result of extensive discussions within the Internet community. Ex. 1048 (RFC 1422) at 1-3.

A157. RFC 1422 describes a certificate authority infrastructure based on X.509 that was designed to provide authentication services. *See* Ex. 1048 (RFC 1422) at 6-7, 20-22. While the infrastructure was designed for supporting PEM services, it was a generic infrastructure that could be used for many purposes. Ex. 1048 (RFC 1422) at 1-2 (“This document goes beyond X.509 by establishing procedures and conventions for a key management infrastructure for use with Privacy Enhanced Mail (PEM) and with other protocols, from both the TCP/IP and OSI suites, in the future.”).

A158. The certificate authority technique described in RFC 1422 was incorporated into copy protection and distribution method. For example, in his article presented at the IMA conference, Kahn described using the PEM certificate structure to provide authentication and notarization services. As Kahn explained:

One of the basic problems with this application of public key cryptography is knowing whether the public key found in the directory for a given correspondent is really that correspondent's key or a bogus one inserted by a malicious person. **The way this is dealt with in the Privacy- Enhanced Mail system is to create certificates containing the name of the owner of the public key and the public key itself, all of which are digitally signed by a well- known issuing authority.** The public key of the issuing authority is widely publicized so it is possible to determine whether a given certificate is valid.

Ex. 1019 (IMA – Kahn) at 116-117 (emphasis added).

A159. Use of certificate authorities in copy protection schemes also was described in Rosen. The Rosen system employed a hierarchy of certificate authorities to provide authentication services. This is shown in Figure 6A:

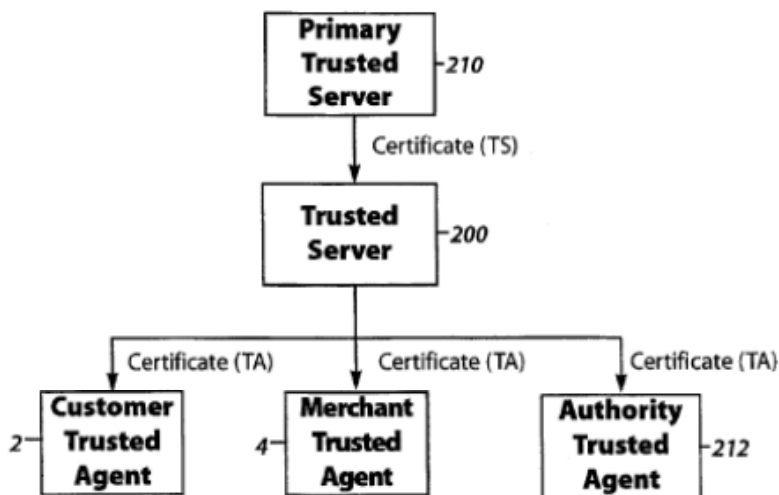


Figure 6A

Ex. 1020 (Rosen) at Fig. 6A; see *id.* at 10:51-11:13 (“FIG. 6A shows the system security hierarchy. At the top of the hierarchy, and located at the Trusted Agency Network 208, are primary trusted servers 210 which certify and provide trusted server certificates (cert(TS)) to all the trusted servers 200 in the system.”).

A160. In the Rosen system, the certificates would be used in the process of establishing secure sessions between two trusted agents. Ex. 1020 (Rosen) at 37:19-38:20; see *id.* at Figs. 38A-38E.

V. SECTION B: General Issues Related to My Patentability Analysis

B1. As I explain in more detail below, I believe the claims of the Stefik patents would have been considered obvious by a person of ordinary skill in the art based on a number of prior art references, particularly when the claims are given their broadest reasonable interpretation consistent with the specification.

A. Overview of the Stefik Patent Specification

B2. The six Stefik patents share a similar, though not identical, specification. I discuss some of these differences in my separate declaration regarding the '160 patent (*i.e.*, Ex. 1015).

B3. Five of Stefik patents (*i.e.*, the '859, '956, '072, '576 and '007 patents) have a specification that is more or less identical except for a portion called the "Summary of the Invention." The drawings in these five patents are also identical. Exhibit 1060 is a table showing the different Summary of the Invention sections in the five patents. I will refer primarily to this identical part of the five Stefik patents in my discussions below.

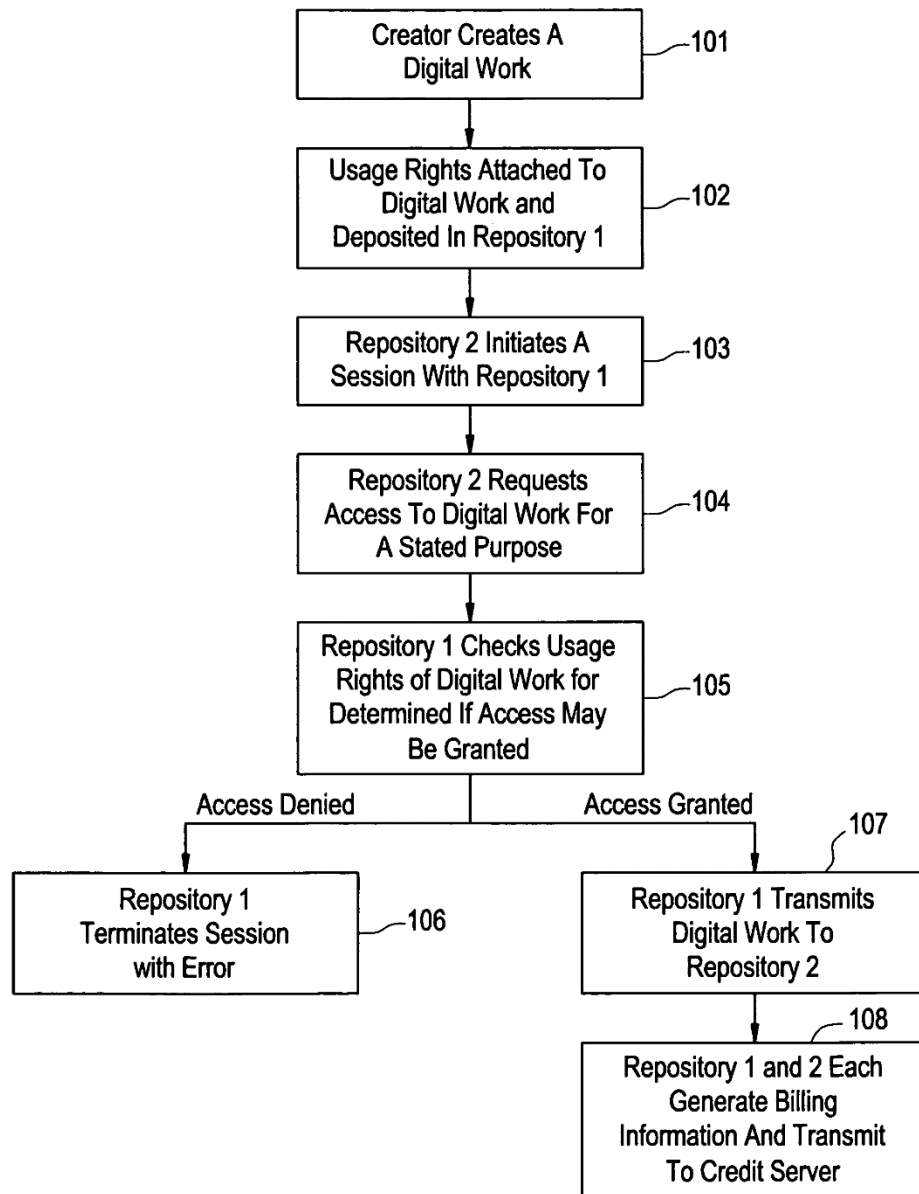
B4. At a general level, the Stefik patents relate to distributing digital works and enforcing usage rights associated with each work. Ex. 1001 (859) at 1:11-12, 6:33-65, Fig. 1.

B5. The Stefik patents explain that when digital content is introduced into the Stefik system, a set of usage rights is permanently attached to the content to

form what Stefik calls a “digital work.” Ex. 1001 (859) at 6:11-16 (“A key feature of the present invention is that usage rights are permanently ‘attached’ to the digital work. Copies made of a digital work will also have usage rights attached. Thus, the usage rights and any associated fees assigned by a creator and subsequent distributor will always remain with a digital work.”), 10:45-56, 49:52-54, 50:8-12.

B6. Examples of digital content include software and a photograph. The Stefik patent shows that these usage rights remain with the digital as it is distributed through the Stefik system. Ex. 1001 (859) at 6:33-65 (“Referring to FIG. 1, a creator creates a digital work, step **101**. The creator will then determine appropriate usage rights and fees, attach them to the digital work, and store them in Repository **1**, step **102**. . . .”), 10:45-56, 49:52-54, 50:8-12.

FIG. 1



B7. A set of trusted devices called “repositories” are used to store digital works, control access to digital works, and enforce the usage rights of digital works. Ex. 1001 (859) at 6:17-21 (“The enforcement elements of the present invention are embodied in repositories.”); *see also* Ex. 1003 (072) at 4:26-27

(“Digital works are stored in repositories. Repositories enforce the usage rights for digital works.”).

B8. Repositories are the only devices described as “trusted” in the Stefik specification. Ex. 1001 (859) at 7:32-37; *id.* at 6:29-32 (“Thus, the digital work genie only moves from one trusted bottle (repository) to another . . .”); *id.* at 7:2-4 (“It is fundamental that repositories will share a core set of functionality which will enable secure and trusted communications.”); 6:22-25 (“The combination of attached usage rights and repositories enable distinct advantages over prior systems.”).

B9. Repositories also are the only devices in the Stefik scheme that enforce usage rights. *E.g.*, Ex. 1001 (859) at 6:12-16, 6:17-21 (“The enforcement elements of the present invention are embodied in repositories. Among other things, repositories are used to store digital works, control access to digital works, bill for access to digital works and maintain the security and integrity of the system.”).

B10. The Stefik patents explain that digital works with “attached usage rights” and “repositories” are the two key features of the system. *Id.* at 6:22-32 (“The combination of attached usage rights and repositories enable distinct advantages over prior systems.”). As the patents explain:

[I]n the prior art, payment of fees are primarily for the initial access.

In such approaches, *once a work has been read, computational control over that copy is gone*. Metaphorically, “*the content genie is out of the bottle* and no more fees can be billed.” In contrast, *the present invention never separates the fee descriptions from the work*.

Thus, the digital work genie only moves from one trusted bottle (repository) to another, and *all uses of copies are potentially controlled and billable*.

Ex. 1001 (859) at 6:22-32 (emphases added); *id.* at 6:13-16 (“the usage rights and any associated fees assigned by a creator and subsequent distributor *will always remain* with a digital work”).

B11. A person of ordinary skill reading the Stefik patents would understand that the successful operation of the Stefik system requires these two key elements (*i.e.*, attached usage rights and repositories). If either is removed (*i.e.*, if a digital work is transmitted to a computer that is not a repository or if the usage rights are detached from the content), then control over the digital work is lost and there is a risk that users will be able to reproduce and redistribute the contents without limitation.

B12. The bulk of the Stefik patents focuses on a usage rights language that was designed to control all aspects of the distribution and use of each digital work. Ex. 1001 (859) at 9:47-10:56, 11:43-50, 16:63-25:36. A person of ordinary skill reading the Stefik patent would understand that what Stefik was really focused on

was creating a universal usage rights language that could be used to define every possible way digital content could be used or distributed.

B13. The security features described in the Stefik specification are conventional security technologies. These include tamper-resistant hardware, digital certificates, digital signatures, and secure communications protocols. *E.g.*, Ex. 1001 (859) at 11:64-67 (physical integrity), 12:24-33 (standard cryptography techniques), 12:36-43 (code signing), 13:1-9 (digital certificates), 13:16-20 (tamper-resistant processor), 26:1-9 (public key encryption).

B14. The concept of “attaching” usage rights to content also was well-known in 1994. File systems have been attaching usage rights to stored files for many decades, to restrict access only to the user that created or owns the file, and to those given access rights by the user. *See ¶¶ A91-A97, above.*

B15. As I explain below, each of the technologies used in the Stefik patent was well-known by 1994.

B. The Claims of the Stefik Patents I Am Addressing in this Report

B16. In this report, I am addressing a set of claims from each of the six Stefik patents. The claims I address are summarized in the following table:

<u>Patent</u>	<u>Claims</u>
'859 Patent	1, 3, 8, 13, 15, 16, 19, 20, 21, 24, 29, 31, 33, 36, 40, 42, 43, 46-48, 51, 58, 60, 62, 65, 69, 71, 72, 75, 76, 81
'072 Patent	1, 8, 10, 16, 18, 24
'956 Patent	1-18

'007 Patent	1-15
'576 Patent	1, 2, 4, 7, 8, 9, 10, 15, 18, 19, 21, 24, 25, 26, 27, 29, 32, 34

B17. The text of these claims is reproduced in claim charts following my analysis of each prior art reference.

C. Interpretation of Certain Claim Terms

B18. Each of the challenged patents claims priority to an application filed on November 23, 1994. I understand that when the Board issues a final written decision in these proceedings, the challenged claims will be expired.

B19. I understand that in an *inter partes* review proceeding involving the claims of an expired patent, claim terms are given their ordinary and customary meaning, as would be understood by a person of ordinary skill in the art, at the time of the invention, in light of the language of the claims, the specification, and the prosecution history of record.

B20. I also understand that, where a patent applicant explicitly defines a term to mean something in the patent disclosure, that definition should be used when evaluating the claims. An explicit definition will be something like “a ‘foo’ means ‘a widget that is 4 inches wide by 6 inches long.’”

B21. I understand that if no explicit definition is provided for a term in the patent specification, the claim terms must be given their plain meaning unless that would be plainly inconsistent with how the term is being used in the claim or the patent specification.

B22. I also understand that the Patent Owner has argued that certain terms in the claims have a particular meaning in district court litigation and in other PTO proceedings. I have considered those interpretations in reaching my conclusions about what the terms in the claims mean.

B23. I understand that when considering the claims of an expired patent, the standards used by the PTO are the same as those used by a district court.

B24. The Stefik patents share a similar (though different) specification, and the claims of each patent use many of the same terms.

B25. For each term construed below, I identify the patents in which the term appears. For ease of reference, my citations to the specification will all be to the '859 patent, unless otherwise noted.

B26. Each specification contains a “glossary,” which defines certain terms used throughout the patents and claims. *E.g.*, Ex. 1001 (859) at 49:30-51:14.

1. “Usage Rights” and “Manner of Use”

B27. The terms “usage rights,” “usage right,” or “usage rights information” is used in every independent claim of the Stefik patents.

B28. The '859 and '072 patent claims further provide that the “usage right” or “rights” specify at least one “manner of use” of the content.

B29. The glossary of the '859, '576, '072, '956, and '007 patents contain a definition of the term “usage rights”:

Usage Rights: A language for defining the **manner** in which a digital work may be used or distributed, **as well as any conditions** on which use or distribution is premised.

Ex. 1001 (859) at 51:7-10 (emphasis added).

B30. The glossary defines usage rights as a “language” for defining how a digital work may be used or distributed. Describing “usage rights” as a language indicates that Stefik believed his actual contribution was to create a “usage rights language” that could be used to specify all the different ways content could be used or distributed. This is consistent with way the Stefik patents describe the technical aspects of its systems – it points to conventional technologies, such as digital certificates, digital signatures, public key and symmetric encryption, etc. It also describes the elements of its system and data constructs in conceptual terms. This opinion is based on my knowledge and experience, my understanding of the state of the art in 1994, and the Stefik specification itself. *See also* ¶ B12, *above*.

B31. The glossary definition of “usage rights” indicates that “usage rights” contain two different elements.

B32. The first element in the usage right is specifies a “manner of use” that indicates how content can be used or distributed. The “manner of use” correspond to a “right” to take a particular action with a digital work. This element is mandatory – every usage right must contain a “manner of use” value.

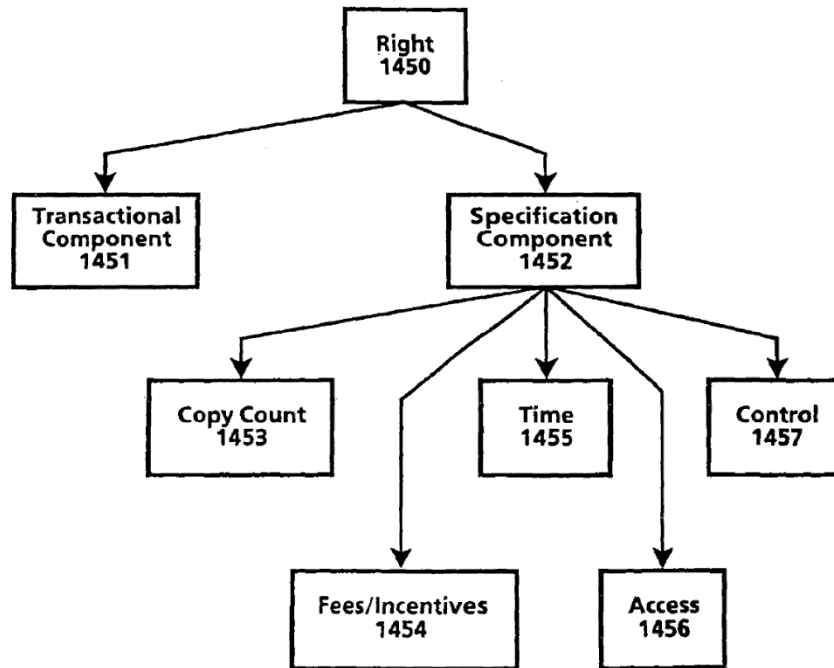
B33. The second element in the usage right is a “condition” on that manner of use, which would correspond to a limitation (*e.g.*, how, when, by whom) on that manner of use. This second element is optional.

B34. While the ’859, ’576, ’956, and ’007 patent claims use the phrase “usage rights”, I note that in the ’072 patent claims say “at least one usage right.” *Compare* Ex. 1001 (859) at 51:26-30 (’859 claim 1 – “usage rights associated with the content”) *with* Ex. 1003 (072) at 52:10-11 (’072 claim 1 – “at least one usage right associated with the digital document”). Although most of the claims specify “usage rights” (plural), it is helpful to analyze first what a “usage right” (singular) is.

B35. Throughout the specification, a usage right is described as consisting of the same two elements identified in the glossary definition: a “right” element (or manner of use) and a “conditions” element. *E.g.*, Ex. 1001 (859) at 6:3-7, 9:47-54, 17:23-25, 18:21-25, Fig. 10. While the specification uses varying terms to refer to these two components, it always depicts a “usage right” as containing these two elements.

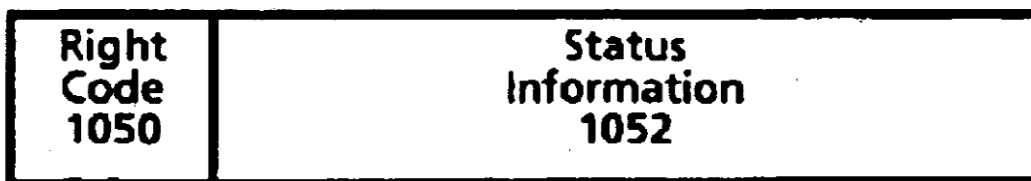
B36. Two figures depict a “usage right,” and each shows that the two elements are conceptually different in their roles – one is defining an action (*e.g.*, manner of use, transactional component, right code) and the other is defining a

condition on taking that action (*e.g.*, condition, specification component, status information). Figure 14:



B37. Figure 10 shows this same two element structure:

Figure 10

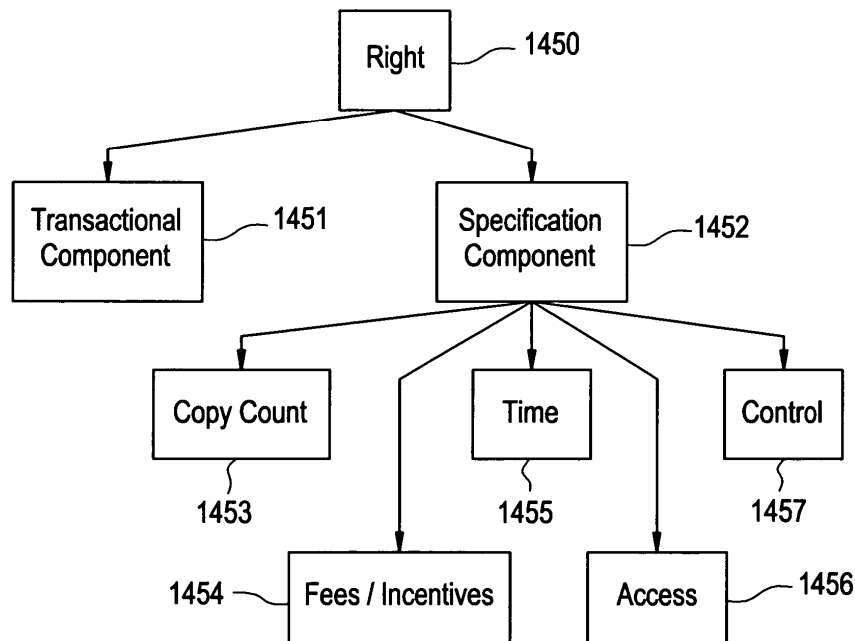


B38. The Stefik patents describe the “right” element (also called “the transactional component”) as data that identifies a particular manner of using

content or of distributing content. *E.g.*, Ex. 1001 (859) at 17:25-32 (“The transactional component 1451 corresponds to a particular way in which a digital work may be used or distributed.”). The specification makes clear that “[e]ach usage right *must* specify a right code,” *id.* at 18:21-23, which corresponds to the particular “use or distribution privilege[] [] embodied by the right,” such as “COPY or PRINT,” *id.* at 17:25-28.

B39. The way that the usage rights concept is described in the Stefik patents also makes clear that each usage right will contain one “right code” that defines one “manner of use.” Each usage right can specify multiple conditions on that manner of use. This is illustrated in Figure 14 which shows the single “manner of use” and multiple “conditions” in the usage right.

FIG. 14



B40. The Stefik patents contain numerous examples of “rights,” and each one shows a right as an affirmative action that can be taken with respect to the work. Several examples of rights are “play,” “print,” “copy,” “transfer,” “loan,” “backup,” and “install.” *Id.* at 18:35-19:52.

B41. The Stefik patents envision creation of a language for a digital work that will reflect all possible uses of the contents in that digital work, with the right code identifying a specific right from a “catalog of possible rights” that can be taken with respect to a digital work. Ex. 1001 (859) at 18:35-41; *see id.* at Figure 15.

B42. The Stefik patents describe the “conditions” element (also called “specifications component” or “status information”) of a usage right as providing limitations or preconditions on the right. Ex. 1001 (859) at 18:21-25, 17:23-38 (“The specifications components 1452 are used to specify conditions which must be satisfied prior to the right being exercised or to designate various transaction related parameters.”); *see id.* at 10:1-29 (Table 1).

B43. The Stefik patents show that the conditions element is distinct from the right element. The Stefik patents enumerate all the different types of rights, (Ex. 1001 (859) at 18:35-20:8), and then it separately lists all the different conditions that can be placed on a right, (*id.* at 20:9-24:10). For example, the specification explains that the “copy-count” condition “defines the number of ‘copies’ of a work subject to the right.” *Id.* at 20:15-17.

B44. Each condition must be associated with a particular right in the Stefik system, which means that a “usage right” cannot contain only a condition. *See, e.g., id.* at Figure 14 (shown above). For example, “The copy-count is associated with each right, as opposed to there being just a single copy-count for the digital work.” Ex. 1001 (859) at 20:18-20; *id.* at 20:46-47 (“It is often desirable to assign a start date or specify some duration as to **when a right** may be exercised.”). This can be clearly seen with respect to the “Usage Rights Grammar” described in the specification, which shows each “usage right” must be in the following format:

“Grammar element 1502 ‘Right:=(Right-Code{Copy-Count}{Control-Spec}{Time-Spec}{Access-Spec}{Fee-Spec})’.” *See* Ex. 1001 (859) at 18:19-25; *see also id.* at 17:41-50.

B45. Also, the Stefik patents show that the “right code” identifies the particular right, and then identify a list of all possible conditions on the right. As the Stefik patents explain:

Each usage right must specify a right code. Each right may also optionally specify conditions which must be satisfied before the right can be exercised. These conditions are copy count, control, time, access and fee conditions.

Ex. 1001 (859) at 18:21-25.

B46. This distinction between “rights” (also “manner of use”) and “conditions” also is reflected in the Stefik claims.

B47. For example, dependent claim 23 of the ’859 patent specifies “wherein the usage rights include at least one condition that must be satisfied to exercise the manner of use.” Ex. 1001 (859) at 52:24-26.

B48. Dependent claim 5 of the ’072 patent specifies “wherein the at least one usage right also specifies one or more conditions which must be satisfied before the manner of rendering may be exercised.” Ex. 1003 (072) at 52:32-34.

B49. From all of these explanations, it is clear that a “manner of use” within the “usage right” cannot be the same data element as a “condition” within a

“usage right.” In other words, as described in the ’072 patent, the usage right could not store a single token which a repository understands to mean “play” and then use that single token both to define the “manner of use” and a condition on exercising that manner of use. Instead, in the Stefik scheme, there would be a first token or value encoding the “manner of use” in the usage right, and then a second or third (etc.) token or value that defines the condition or conditions on exercise of that right.

B50. As I mentioned above, the glossary defines “usage rights” as a “language” for specifying how a digital work may be used or distributed. In some embodiments, each “usage right” is shown as a statement in a “language” in the traditional computer science sense – a defined set of keywords having a particular grammar for forming statements. Stefik calls the makeup of that language a “Usage Rights Grammar.” *E.g.*, Ex. 1001 (859) at 18:9-10; *see also id.* at 17:9-22:43 and Fig. 15.

B51. In other embodiments, a “usage right” is shown as a set of data that encodes a particular right and any conditions thereon. *E.g.*, Ex. 1001 (859) at Figs. 10 & 14. Even where a “usage right” is shown as a set of data, those data are generated by taking statements in a language and “encod[ing] [them] in a suitable form” (*e.g.*, a numeric code that can be embedded in the work), and those data are capable of representing all the possible ways the digital work can be used within

the system. *E.g., id.* at 9:47-57 (“The right code field 1001 will contain a unique code assigned to a right.”), 17:6-8 (“Once the usage rights statements are generated, they are encoded in a suitable form for accessing during the processing of transactions.”).

B52. Whether “usage rights” are shown as statements in a grammar or a set of data encoding such statements, every depiction of usage rights in the specification shows a single “usage right” as containing data representing the right (or “manner of use”) and separate data representing all the conditions (if any) on exercising the right. *Id.* at 16:66-17:3.

B53. For the Stefik systems to work as they are described in the Stefik patents, each usage right must contain all the information the repository needs to determine whether the requested use of the content is permitted; otherwise repositories would not be able to interpret the right to determine if a particular use was permitted and if the conditions had been satisfied. *See id.* As I explain further below (§ V.C.2), the Stefik patents say that repositories contain the “enforcement” elements of its system and use the information in “usage rights” to carry out that enforcement function.

B54. The Stefik patents show that it is possible to attach multiple “usage rights” to a single copy of a digital work, each with its own set of conditions on “exercising” the right. This is how the Stefik patents show multiple “manners of

use” being attached to a digital work. In other words, in the Stefik scheme, it is possible to have a 1:1 or a 1:many relationship between the digital work and usage rights.

B55. The converse, however, is not true. In the design of the Stefik system, is not possible for a single instance of a usage right to be “shared” with multiple digital works. Because each usage right is attached to a particular digital work, a single instance of a “usage right” applies to that single digital work. While different works could have equivalent rights (*e.g.*, the rights to print and transfer), each work would have its own instance of those rights. There is no method disclosed in the Stefik patent showing how a single instance of a right could be attached to multiple digital works.

B56. For example, the Stefik patent refers to something it calls a “composite digital work.” This is defined in the glossary as being:

A digital work comprised of distinguishable parts. ***Each of the distinguishable parts is itself a digital work which has usage rights attached.***

Ex. 1001 (859) at 49:48-51.

B57. The composite digital work in the Stefik scheme is basically a set of individual digital works, with each of the works having an attached usage right. In other words, the Stefik patents show that the usage rights attached to each component digital work controls the use of that component.

B58. Multiple usage rights may be attached to one digital work (each specifying a particular manner of use and associated conditions). Where a user is given permission to take multiple actions with a particular piece of content, multiple usage rights will be attached to it.

B59. Therefore, the proper construction of a “**manner of use**” is “a defined way of using or distributing a digital work (for example, PLAY, COPY, or PRINT), as distinct from conditions which must be satisfied before that way of using or distributing the digital work is allowed.”

B60. Therefore, the proper construction of “usage rights” is “statements in a language for defining the manner in which a digital work may be used or distributed, as well as any conditions on which use or distribution is premised. Usage rights must be permanently attached to the digital work.”

B61. Consistent with these constructions, in an implemented Stefik system, each “usage right” will take the form of data permanently attached to a particular copy of a digital work that specifies (i) one “right” or “manner of use” out of a defined set of manners of use that may be taken with that copy of the digital work and (ii) optionally includes data defining conditions associated with that manner of use. These data contain the “statements” of the language or data encoding such statements, and they are distinct from data that is used by a repository to enforce the “usage right” (*e.g.*, encryption keys, digital certificates, digital signatures).

2. “Repository”

B62. The term “**repository**” is explicitly used in the independent claims of the ’072, ’859, and ’576 patents. In addition, a repository is required by the independent claims of the ’956 and ’007 patents, as I explain below with respect to the phrase “determining . . . if the recipient device can be trusted.”

B63. The glossary in the specification defines a repository as

Conceptually a set of functional specifications defining core functionality in the support of usage rights. A repository is a trusted system in that it maintains physical, communications and behavioral integrity.

Ex. 1001 (859) at 50:48-52.

B64. The glossary provides that a repository must “support [] usage rights.” By support usage rights, the definition means “enforce usage rights.” The specification explains that “[t]he enforcement elements of the present invention are embodied in repositories.” *Id.* at 6:17-18. In fact, repositories are the only systems shown as enforcing usage rights.

B65. The glossary definition provides that a repository is a “trusted” system, and that “trusted” systems are ones which possess “physical, communications and behavioral integrity.” Ex. 1001 (859) at 50:48-52. The specification also explains that, for any device to be a “trusted” system, it must possess three types of integrity. *Id.* at 11:62-12:50.

B66. Physical integrity means the repository protects access to the repository and digital works by “never allow[ing] non-trusted systems to access the digital works directly.” *Id.* at 11:67-12:5.

B67. Communications integrity means that a repository only communicates with other devices that are able to present proof they are certified repositories, and they use exchange of digital certificates, encryption, and nonces to protect communications. *Id.* at 12:24-33.

B68. Behavioral integrity means software must include a digital certificate to be installed in the repository. *Id.* at 12:40-43, 12:46-50.

B69. Both communications and behavioral integrity require “digital certificates.” The glossary defines a digital certificate (an “Identification (Digital) Certificate”) as “[a] signed digital message that attests to the identity of the possessor.” Ex. 1001 (859) at 50:16-19; *see id.* at 26:54-60.

B70. A person of ordinary skill in the art would understand that the “identification certificate” is a standard digital certificate.

B71. A digital certificate is a signed message that binds an identity to a public key. Ex. 1017 (Denning) at 169-70; *see* ¶¶ A144 to A159, *above*. Digital certificates typically are issued by a certificate authority, and the authority’s digital signature on the certificate attests that the entity identified in the certificate is associated with the identified public key. *Id.*

B72. The specification explains that, in the context of behavioral integrity, the digital certificate must be used to confirm that application software has been certified (as genuine). Ex. 1001 (859) at 12:40-43 (the digital certificates provides “proof of [the software’s] certification”). For the repository to achieve behavioral integrity, the software installed into the repository must be signed. Absent a digital signature of the software, any entity could simply copy and append a digital certificate to a software application, making it appear that the software were certified.

B73. A person of ordinary skill in the art would have understood that for the software to be “certified,” it would need to include the digital signature of the entity certifying it. The digital signature would validate the integrity of the software (*i.e.*, that it has not been changed since the entity signed it) and it would authenticate that entity signing the software was its source. *See* ¶¶ A98 to A100, *above* (describing digital signatures); *see also* ¶¶ A144 to A159, *above*.

B74. Stefik describes this process as “certifying” software, but more common terms for this process are “code signing” or “signing software.” Code signing has long been a standard practice for ensuring the integrity and authenticity of distributed software. *See, e.g.*, Ex. 1017 (Denning) at 15-16; Ex. 1033 (Davida) at 314-15. The signature can be checked any time the code is loaded into the system, such as installation or prior to execution. Ex. 1033 (Davida) at 317 (“A

program can be checked in its [sic] entirety when it is loaded into the system and in addition, periodically by some background task.”).

B75. In this process, the digital certificate, or more precisely the public key in the digital certificate, would be used to validate the digital signature of the software. If the signature is valid, that means that the entity identified in the digital certificate created the signature, and that the software has not been modified since it was signed. A person of ordinary skill in the art would have understood that the signed software would need to include either the digital certificate itself or information sufficient to allow the recipient to obtain the applicable digital certificate (*e.g.*, from a certificate authority or from a cache of saved certificates).

B76. It would not make sense to distribute software with a digital certificate but not a digital signature of the software. This is because the digital certificate alone cannot provide assurance of integrity or authenticity. Nothing would stop someone from modifying the software or from copying the digital certificate and appending it to another application. The digital signature enables the recipient to verify that the message (*e.g.*, software) being attested to came from the purported source; the digital certificate provides the authenticated public key of the purported source which is needed to verify the signature.

B77. The specification also shows that the digital signature is a necessary component of the software certification process. In the only example of validating

a digital certificate associated with software that is shown, the specification describes the process as follows:

The requester decrypts the digital certificate using the public key of the master repository, recording the identity of the supplier and creator, a key for decrypting the software, the compatibility information, and a tamper-checking code. (This step certifies the software.) The requester decrypts the software using the key from the certificate and computes a check code on it using a 1-way hash function. If the check-code does not match the tamper-checking code from the certificate, the installation transaction ends with an error. (This step assures that the contents of the software, including the various scripts, have not been tampered with.)

Ex. 1001 (859) at 41:45-57. In this passage, the specification is explaining that the master repository digitally signs the software application by encrypting a tamper-checking code (which is a cryptographic hash of the software) and including it with the digital certificate. The repository installing the software computes a hash of the software and checks that against the code from the “certificate.” *See* ¶¶ A137-A142, *above*.

B78. I understand that the Board has interpreted the term repository as “a trusted system, which maintains physical, communications and behavioral integrity, and supports usage rights.” *E.g.*, IPR2013-00137, Paper 58 (July 1, 2014) (“ZTE 859 Final Decision”) at 8.

B79. I understand the Board has construed “physical integrity” as “preventing access to information by a non-trusted system.” *E.g.*, ZTE 859 Final Decision at 9-10.

B80. I understand the Board has construed “communications integrity” as “only communicates with other devices that are able to present proof that they are trusted systems, e.g., by using security measures such as encryption, exchange of digital certificates, and nonces.” *E.g.*, ZTE 859 Final Decision at 10-11.

B81. I understand the Board has construed “behavioral integrity” as “requiring software to include a digital certificate in order to be installed in the repository.” *E.g.*, ZTE 859 Final Decision at 11..

B82. I understand the Board has explained that behavioral integrity means that when a digital work distributed via a repository is software (*e.g.*, a software application that plays music or other content), the digital work is required to have a digital certificate before being installed onto a repository. ZTE 859 Final Decision at 16-17; *see* Ex. 1001 (859) at 12:47-50 (“If the digital certificate cannot be found in the digital work . . . , then the software cannot be installed.”).

B83. I agree with the Board's interpretation of the term repository as being “a trusted system, which maintains physical, communications and behavioral integrity, and supports usage rights.”

B84. The Board’s construction is consistent with the specification – as I explained above, the only example of validating a “digital certificate” associated with software in the ’859 patent is when installing “a digital work as runnable software on a repository.” Ex. 1001 (859) at 41:38-42:6; *see id.* at 19:48-54 (“‘Configuration-Code:=Install|Uninstall’ lists a category of rights for installing and uninstalling software on a repository (typically a rendering repository.) This would typically occur for the installation of a new type of player within the rendering repository. . .”).

B85. I understand that in IPR2013-00137, Patent Owner had argued that behavioral integrity does not apply to digital works that are software, but rather, that it applied only to upgrades to the “system software” that operated the repository (*i.e.*, operated to enforce usage rights). ZTE 859 Final Decision at 15.

B86. Patent Owner’s argument is not consistent with the Stefik patents. While Stefik mentions that repositories may have a functional (*i.e.*, a software-based) embodiment, the Stefik patents do not discuss how that software is installed or updating that software. *See* Ex. 1001 (859) at 13:10-15. The only discussion in the Stefik patents about software relates to software that is packaged into a digital work.

B87. The Board also has explained that behavioral integrity means that software must have a “digital certificate.” ZTE 859 Final Decision at 19. I

understand the Board has rejected Patent Owner's argument that behavioral integrity could be satisfied by any data that "ensures a repository does what it is supposed to do." ZTE 859 Final Decision at 18-19, 23-24.

B88. Therefore, the proper construction of the term "repository" is "a trusted system in that it maintains physical, communications and behavioral integrity, and it supports usage rights by controlling which actions can be taken with a digital work by using information in that work's usage rights." Ex. 1001 (859) at 50:48-52; ZTE 859 Final Decision at 8. Physical, communications, and behavioral integrity should be defined consistent with the Board's definitions in the ZTE 859 Final Decision and with my explanation above.

3. Distributed repository

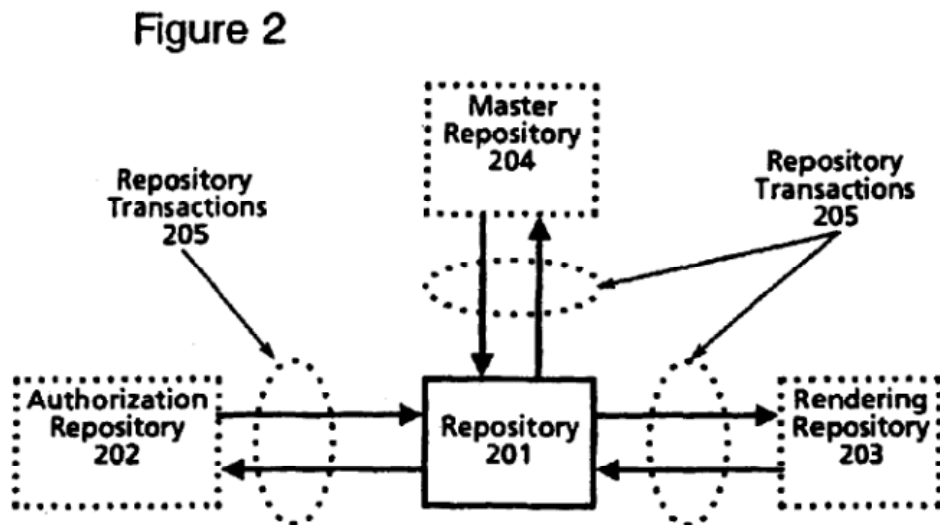
B89. The phrase "**distributed repository**" is used in the claims of the '859 patent. Each independent claim specifies that a "distributed repository" has both a "requester mode" and "server mode" of operation.

B90. The '859 patent specification does not define the term "distributed repository." However, the preamble specifies that the repository is for use in "a distributed system for managing use of content." Ex. 1001 (859) at 51:15-20.

B91. When the term "distributed" is used with respect to a system, a person of ordinary skill in the art would understand that it typically means the functionality of the system is spread across two or more devices. *See* Ex. 10XX

(Microsoft Computing Dictionary Third Ed.) at 154 (“distributed processing: A form of information processing in which work is performed by separate computers linked through a communications network. . . . True distributed processing has separate computers perform different tasks in such a way that their combined work can contribute to a larger goal. . . .”).

B92. Figure 2 of the '859 patent shows a repository as part of a larger system that includes an authorization repository, a master repository, and a rendering repository. A person of ordinary skill in the art would understand that Figure 2 is depicting a simplistic architecture for a distributed system.



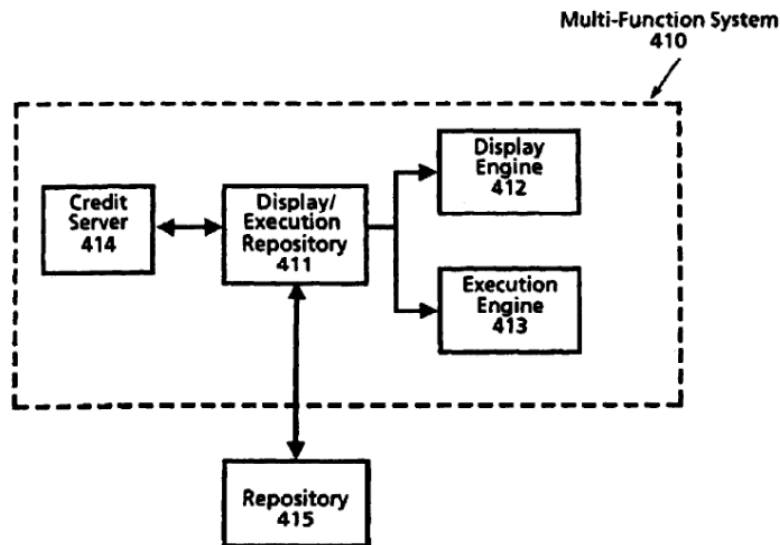
B93. The glossary defines “requester mode” as “[a] mode of repository where it is requesting access to a digital work.” Ex. 1001 (859) at 50:53-55. The glossary defines “server mode” as “[a] mode of a repository where it is processing an incoming request to access a digital work.” *Id.* at 50:60-62. These definitions

are consistent with how the server and requester modes of operation are described elsewhere in the specification. *E.g.*, Ex. 1001 (859) at 7:4-12, 29:38-47, Fig. 2.

B94. When in server mode of operation, the incoming request can be from another repository, or it can be from a rendering device connected to the repository. Ex. 1001 (859) at 29:44-47 (“In many cases such as requests to print or view a work, the requester and server may be the same device and the transactions . . . would be entirely internal”).

B95. Going back to Figure 2, a rendering repository in the requester mode of operation might request access to a work stored on the repository, or it might request a new digital certificate from the master repository or the authorization repository. A rendering repository in server mode would receive a request from a rendering device/engine to render content, and it would enforce any usage rights associated with that content. A rendering repository is shown in more detail in Figure 4b:

Figure 4b



B96. Therefore, the proper construction of the term “**distributed repository**” is “a ‘repository’ capable of operating in server and requester mode, which communicates with other repositories over a network using digital certificates.”

4. **Recipient computing device / recipient apparatus / sending computing device / the at least one recipient computing device has been determined to be trusted to receive the digital content**

B97. The '956 and '007 patent claims specify a “**sending computing device**,” a “**recipient computing device**,” and a “**recipient apparatus**.” The claims further specify that the “sending computing device” determines whether “the at least one recipient computing device [is] **trusted to receive the digital content**.”

B98. The only example of a “trusted” system identified in the specification is a “repository.” The specification defines a “repository” as “a trusted system in that it maintains physical, communications and behavioral integrity.” Ex. 1001 (956) at 50:18-22.

B99. The only uses of the term “trust” in the specification are in association with repositories. For example, the specification explains that

Communication with a master repository 205 occurs in connection with obtaining an identification certificate. **Identification certificates are the means by which a repository is identified as “trustworthy”.** The use of identification certificates is described below with respect to the registration transaction.

Ex. 1001 (956) at 7:43-48 (emphasis added); *id.* at 6:42-44 (“Thus, the digital work genie only moves from one trusted bottle (repository) to another . . .”); *id.* at 7:13-15 (“It is fundamental that repositories will share a core set of functionality which will enable secure and trusted communications”).

B100. I believe a person of ordinary skill in the art would understand that a device that is or has a repository is a trusted device based on how the Stefik patents are written. Thus, the proper construction of “recipient computing device” and “recipient apparatus” is “recipient repository.” Thus, the proper construction of “sending computing device” is “sending repository.”

B101. The Stefik patents explain that the trustworthiness of a device will be determined during the session initiation process. *E.g.*, Ex. 1001 (956) at 7:43-48; *id.* at 6:55-58 (“the session initiation includes steps which help to insure that the respective repositories are trusted.”). The Stefik patents show that this determination is made by validating a device’s identification certificate. *Id.* at 26:54-60 (“A registration message is comprised of an identifier of a master repository, the identification certificate for the repository-1 and an encrypted random registration identifier. **The identification certificate . . . attests to the fact that the repository (here repository-1) is a bona fide repository.**” (emphasis added)); *id.* at 27:28-33 (“Note that rather than terminating in error, the transaction could request that another registration message be sent based on an identification certificate created by another master repository. This may be repeated until **a satisfactory identification certificate is found, or it is determined that trust cannot be established.**” (emphasis added)). The specification does not show any other examples of determining whether a device can be trusted.

B102. A person of ordinary skill in the art would understand that the claimed “determination” of whether a recipient device can be trusted, as it is described in the Stefik specification, can be performed by authenticating the device by validating the device’s digital certificate during the initiation of a session. *See id.* at 7:44-46 (“Identification certificates are the means by which a repository is

identified as ‘trustworthy.’”); *see generally* Ex. 1001 (956) at 26:33-28:43 (describing the “session initiation transaction” for establishing a connection between two repositories).

B103. Therefore, the proper construction of the phrase “the at least one recipient computing device has been determined to be trusted to receive the digital content” means that the recipient computing device has been determined to be a repository by validating a digital certificate associated with the device.

5. Digital content / content / digital work

B104. The terms “**digital content**” (’576, ’956, ’007), and “**content**” (’859) are used in claims of the Stefik patents.

B105. Those terms do not appear alone. “Content” has “usage rights associated with the content.” Ex. 1001 (859) at 51:26-30. “Digital content” is rendered in accordance with “usage rights information.” *E.g.*, Ex. 1005 (956) at 50:57.

B106. The glossary defines “content” as “the digital information (i.e., raw bits) *representing a digital work*.” Ex. 1001 (859) at 49:53-54 (emphasis added). Because “content” is defined to be digital information, there is no difference between content and digital content. Content represents the actual data representing a “digital work.” As I mentioned (§ B5), the Stefik system is used to

distribute content plus attached usage rights (which Stefik calls a “digital work”) and not content alone.

B107. The glossary defines a “digital work” as “Any encapsulated digital information. Such digital information may represent music, a magazine or book, or a multimedia composition. Usage rights and fees are attached to the digital work.”

Ex. 1001 (859) at 50:8-12. A digital work is “content” plus “usage rights.”

B108. The specification explains that “[i]t is **fundamental** to the present invention that the **usage rights are treated as part of the digital work.**” Ex. 1001 (859) at 10:45-46 (emphasis added). Specifically, “[t]he combination of attached usage rights and repositories enable distinct advantages over prior systems”:

As noted in the prior art, payment of fees are primarily for the initial access. In such approaches, once a work has been read, computational control over that copy is gone. Metaphorically, “the content genie is out of the bottle and no more fees can be billed.” **In contrast, the present invention never separates the fee descriptions from the work.** Thus, the digital work genie only moves from one trusted bottle (repository) to another, and **all uses of copies are potentially controlled and billable.**

Ex. 1001 (859) at 6:22-32 (emphasis added).

B109. The specification emphasizes that “usage rights” must “permanently attached to,” *i.e.*, are part of, a digital work:

A key feature of the present invention is that usage rights are permanently “attached” to the digital work. Copies made of a digital work will also have usage rights attached. Thus, the usage rights and any associated fees assigned by a creator and subsequent distributor **will always remain with a digital work.**

Ex. 1001 (859) at 6:11-16 (emphasis added).

B110. Usage rights are attached to digital content the moment digital content is placed in a repository. Ex. 1001 (859) at 6:35-38 (“[A] creator creates a digital work . . . The creator will then determine the appropriate usage rights and fees, attach them to the digital work, and store them in Repository 1 . . .”), Fig. 1. If the digital work transfers from one repository to another, the usage rights will transfer with the work. Ex. 1001 (859) at 6:38-61 (“[t]he digital work remains securely in Repository 1 until a request for access is received. . . . If access is granted, repository 1 transmits the digital work to repository 2 . . .”).

B111. The specification does not disclose repositories being used to store or transfer content without attached usage rights. *E.g.*, Ex. 1001 (859) at 6:28-32 (“the present invention **never separates** the fee descriptions from the work” (emphasis added)). Any content stored in a repository will have usage rights attached to it. Ex. 1001 (859) at 6:35-38, Fig. 1.

B112. The specification shows that a digital work can be any type of content. Ex. 1001 (859) at 5:64-6:3, 18:49-53. For example, it shows that a digital work

can be software programs. Ex. 1001 (859) at 8:17-20. It also shows a digital work can be a photograph. Ex. 1001 (859) at 8:17-20.

B113. A person of ordinary skill in the art would understand the claim terms “content” and “usage rights associated with the content” refer to the two components of a digital work, and that they must be stored in a repository.

B114. Therefore, the proper construction of the terms “content” and “digital content,” is “data (*i.e.*, ‘digital information’) representing a ‘digital work’ whose use or distribution is regulated by repositories using usage rights permanently attached to that instance of the digital work.” Ex. 1001 (859) at 49:51-53, 50:8-12, 10:45-46.

B115. A “digital work” is “Any encapsulated digital information. Such digital information may represent music, a magazine or book, or a multimedia composition. Usage rights and fees are permanently attached to the digital work.” Ex. 1001 (859) at 50:8-12, 10:45-46.

6. Digital document

B116. The term “**digital document**” (072) is used in claims of the ’072 patent. The claims specify that a “digital document” is associated with “at least one usage right.” Ex. 1003 (072) at 52:10-11.

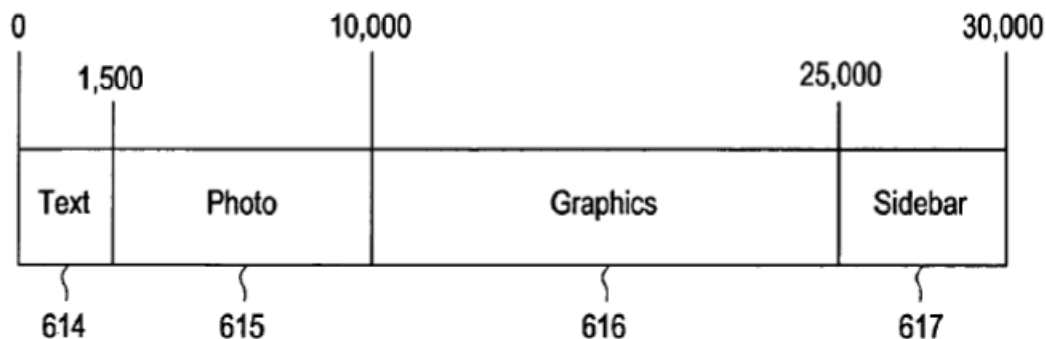
B117. The specification does not use the phrase “digital document.”

B118. The abstract describes a digital document as follows:

A method . . . for securely rendering digital documents, including *storing a digital document* in a document platform; and *storing a usage right* associated with *the digital document*. . . . The *digital document comprises plural parts of digital content*. The usage right includes *plural usage rights respectively associated with each of the plural parts of digital content*. Whether *one of the parts* of the digital document *may be rendered* by the document platform *is determined based [sic] a respective usage right*. If the respective usage right allows [it] to be rendered . . . , the corresponding part...is rendered . . .

B119. The abstract describes a “digital document” as a compound document, that contains distinct documents or works within it. It appears to be describing the documents depicted in Figures 5 and 6 of the ’072 patent, which embody a magazine or a newspaper.

FIG. 6



B120. The specification describes an analogous concept, but refers to it using the term “composite digital work,” which the glossary defines as “[a] digital

work comprised of *distinguishable parts*. *Each* of the distinguishable parts *is itself a digital work which has usage rights attached.*” See *id.* at 50:29-32.

B121. The ’072 patent explains that attaching usage rights to each part of a work enables more granular control over its use and distribution. *Id.* at 17:64-66, 8:46-51 (“Each of the articles and photographs may represent a node . . . *controls, i.e. usage rights, may be placed on each node* enabling control and fee billing to be associated *with each node*”).

B122. The most logical way to interpret the term “digital document” is that it is a type of digital work that Stefik calls a “composite digital work” because a “digital document” (a) it contains two or more “distinguishable” parts, (b) has different usage rights attached to each of the parts, and (c) the different usage rights attached to each part can be independently enforced. *Id.* at 50:29-32.

B123. Each part of a “digital document” therefore would be a “digital work,” which I discussed above. See ¶¶ B104 to B115, *above*.

7. “**Identification certificate**” / “**digital certificate**”

B124. The term “**identification certificate**” is used in the claims of the ’956, ’007, and ’859 patents.

B125. While the claims of the Stefik patents do not explicitly recite the term “**digital certificate**,” many of the security features described in the Stefik patent use “digital certificates” for various reasons.

B126. The glossary in the specification of the '859 patent defines “identification (digital) certificate” as “[a] signed digital message that attests to the identity of the possessor.” Ex. 1001 (859) at 50:16-19. The glossary provides that the certificate “typically” is “encrypted in the private key of a well-known master repository,” which authenticates that the public key in the certificate belongs to the entity identified in the certificate. *Id.*; *id.* at 26:54-60.

B127. A person of ordinary skill in the art would understand that the “identification certificate” is a standard digital certificate. A digital certificate is a signed message that binds an identity to a public key. Ex. 1017 (Denning) at 169-70; *see* ¶¶ A144 to A150, *above*. Digital certificates typically are issued by a certificate authority, and the authority’s digital signature on the certificate attests that the entity identified in the certificate is associated with the identified public key. *See* ¶¶ A149 to A160, *above*.

B128. The specification uses the term “identification certification” to mean a digital certificate issued by a certificate authority. For example, in the context of communications integrity, the '859 patent explains that communications are secured through the exchange of digital certificates. Ex. 1001 (859) at 12:29-33. In this instance, the patent is using the term “digital certificate” synonymously with “identification certificate.” As I explained above, after the certificates are

exchanged between parties to a communication, the parties will use the data in the certificates to authenticate each other. *See* ¶¶ A138 to A152, *above*.

B129. Therefore, the proper construction of “identification certificate” is “a signed digital message that attests to the identity of the possessor,” and it would encompass a digital certificate issued by a certificate authority.

8. Rendering system / video system

B130. Claim 1 of the ’859 patent recites the term “**rendering system.**” The specification of the ’859 patent defines “rendering system” as “[t]he combination of a rendering repository and a rendering device.” Ex. 1001 (859) at 50:43-47; *see also id.* at 50:37-42 (describing “typical” features of a “rendering repository”).

B131. The glossary definition is consistent with how the phrase is used in the claims. The definition also provides that examples of rendering systems include “printing systems, display systems, general purpose computer systems, video systems or audio systems.” Ex. 1001 (859) at 50:43-47.

B132. Therefore, the proper construction of the term “rendering system” encompasses a general purpose computer.

B133. Claims 13 and 69 ’859 patent recite the term “**video system.**” The glossary lists a “video system” as a type of rendering system, but that phrase is not used anywhere else in the specification. *See* Ex. 1001 (859) at 50:43-47.

B134. The specification shows that an execution system can render software programs, a display system can render images such as photographs (*id.*), and a printing system can print out documents, (*id.* at 7:58-61, 7:28-31, Fig. 4a).

Therefore, a “video” system is a system that can display video content.

B135. As Stefik explains, a computer system is a “multifunction device.” *Id.* at 8:16-31, Fig. 4b. A person of ordinary skill in the art would have understood a computer can be used to play video. Therefore, in view of the language of the claims, the specification, and the prosecution history, the proper construction of the term “video system” encompasses a computer that can play video content.

9. Rendering device / execution device / rendering engine

B136. Claims 1, 29, and 58 of the ’859 patent recite the term “**rendering device**,” and claims 16, 43, and 72 recite the term “**execution device**.”

B137. The independent claims of the ’576 patent specify a “**rendering engine**” configured to render digital content. The specification uses the terms “rendering device” and “rendering engine” as synonyms. For example, Figure 4a shows “Print Device 403”, which is described in the specification as “a print device 403” (Ex. 1001 (576) at 8:27-29) and “the print engine 403” (*id.* at 8:36-38). Similarly, Figure 4b depicts “Display Engine 412” and “Execution Engine 413”, which are described as “display device 412” and “execution device 413.” *Id.*

at 8:60-61. Therefore, a “rendering engine” is the same thing as a “rendering device.”

B138. The specification does not define the term “**rendering device**” or “**execution device**.” The specification shows that where a computer system is the rendering system, the computer can act as “a ‘multifunction’ device” that contains both an execution device and a display device. Ex. 1001 (859) at 8:16-31, Fig. 4b.

B139. The specification shows that the computer can render different types digital works or content such as “software programs” and “digitized photographs.” Ex. 1001 (859) at 8:16-31.

B140. It is generally understood that word “rendering” means a process for displaying data in a human-perceptible or usable form, such as processing a digital file into an image or text for display on a video monitor or printing data onto paper. It is generally understood that “rendering” software means executing data on a computer that will cause the programmed computer to perform operations specified by the software.

B141. Therefore, the proper constructions of the terms “**rendering device**” and “**rendering engine**” encompass a computer system that can “render” content. The proper construction of “**execution device**” encompass a computer system that can “execute” software.

10. Document Platform

B142. The phrase “**document platform**” is used in claim of the ’072 patent. Other than in the Abstract and claim language, “document platform” does not appear in the Stefik patents, although a similar term is used once: “Transactions occur between two repositories (one acting as a server), between a repository and a **document playback platform** (e.g. for executing or viewing) . . .” Ex. 1003 (072) at 26:25-27 (emphasis added); *see id.* at 10:52-54.

B143. The claims of the ’072 patent specify that the document platform stores a digital document and at least one usage right associated with the document – which together would be a “digital work.” *See* ¶¶ B106, B113-B116, *above*. They also specify that the digital work is transmitted to the document platform by a “document repository”. Claims 1 and 18 also specify that the document platform enforces usage rights. As I explained above, repositories are the only devices shown in the specification that store digital works, transmit or receive digital works, and that enforce usage rights. Therefore, to the extent “document platform” has any discernible meaning in the ’072 patent, the proper construction of “document platform” is “document repository.”

B144. As I explained above, repositories are the only devices shown in the specification that store digital works, transmit or receive digital works, and that enforce usage rights. *See* ¶¶ B98 to B102, *above*. Therefore, the proper construction of “document platform” is “document repository.”

B145. In IPR2013-00133, the Board found that the broadest reasonable construction of the term “**document platform**” is any computing system that holds a digital document, such as software. Ex. 1001 (072) at 26:25-29; *see id.* at 10:52-54. It further explained that “For example, a ‘document platform’ may be a computing system that executes or views software.” IPR2013-00133, Paper 15 at 17. However, I do not believe that this “broadest reasonable construction” would be the ordinary and customary meaning in light of the specification for this term.

11. Authorization object

B146. The phrase “**authorization object**” is used in the claims of the ’956, ’007, and ’576 patents.

B147. The Stefik patents describe an “authorization object” as being “a digital work in a file of a standard format.” Ex. 1001 (956) at 21:64-67, 40:26-29.

B148. The Stefik patents also consistently show the authorization object as a “digital certificate”: “Conceptually, an authorization is a digital certificate such that possession of the certificate is required to gain access to the digital work.” *Id.* at 7:31-33; *id.* at 21:64-67 (“Authorizations themselves are digital works (hereinafter referred to as an authorization object) . . .”).

B149. The Stefik patents also equate “identification certificates” with the terms “authorization objects” or “authorization certificates.” For example, they say:

As a prerequisite to operation, **a repository will require possession of an identification certificate**. Identification certificates are encrypted to prevent forgery and are issued by a Master repository. **A master repository plays the role of an authorization agent to enable repositories to receive digital works**. Identification certificates must be updated on a periodic basis.

Ex. 1001 (956) at 13:14-21 (emphasis added); *see id.* at 30:16-19 (“For example, install, uninstall and delete rights may be implemented to require that a requester have an **authorization certificate** before the right can be exercised.” (emphasis added)); *id.* at 35:59-60 (describing “authorization certificates”); *id.* at 14:24-29 (identification certificates are “required to enable use of the repository”).

B150. I note the Stefik patents use the term “digital certificate” imprecisely to refer to both a digital certificate and a signed message. A digital certificate is used for **authentication** (showing that a user is who he says he is), not authorization (showing that a user has permission to do something). By contrast, a signed message would be used to show **authorization** (*i.e.*, authenticity and integrity of the grant of authorization can be verified using the digital certificate).

B151. Therefore, the proper construction of the term “**authorization object**” is “a digital work having a standard format that can be used by a repository, and which the repository must possess to receive and use a digital work.”

12. Registration message / registration identifier

B152. The Stefik patents do not contain the phrases “**registration message**” or “**registration identifier.**”

B153. In one passage, the Stefik patents describe a “registration message” as being “comprised of an identifier of a master repository, the identification certificate for the repository-1 and an encrypted random registration identifier.” Ex. 1001 (956) at 26:54-57.

B154. The Stefik patents explain that “[t]he registration identifier is a number generated by the repository . . . , is unique to the session . . . , [and] is used to improve security of authentication by detecting certain kinds of communications based attacks.” Ex. 1001 (956) at 26:66-27:1. They also explain that “[w]hen a sending repository transmits a message to a receiving repository, the sending repository . . . includes its name . . . , **a session identifier such as a nonce** (described below), and a message counter in each message.” Ex. 1001 (956) at 26:17-22. The session identifier “is used to guard against various replay attacks to security.” *Id.* at 26:25-29.

B155. A person of ordinary skill in the art would have understood that the “registration identifier” and the “session identifier” are the same thing: a nonce. As the Stefik patents explain, both are random data used to prevent replay attacks – *i.e.*, a nonce. The technique of including random data in a message to prevent replay attacks was well-known in the art. *See* ¶ A153, *above*.

B156. A person of ordinary skill in the art would have understood that the registration identifier could be a random number, a timestamp, or another piece of unique data. Ex. 1017 (Denning) at 175 (“The timestamp T guards against replays of previous keys.”); *see also* Ex. 1041 (Applied Crypto) at 170-72; Ex. 1020 (Rosen) at 15:46-51.

B157. The Stefik patents themselves recognize that a timestamp is a type of “random and variable information.” Ex. 1001 (956) at 27:45-47 (“A nonce is a generated message based on some random and variable information (e.g., the time or the temperature).”).

B158. Therefore, the proper construction of the term “**registration identifier**” should encompass any random or unique data that can be used to prevent a replay attack, such as a random number or a timestamp.

B159. The Stefik patents show that the registration message is used to exchange digital certificates during the initiation of a secure communications channel between two repositories. *See* Ex. 1001 (956) at 26:46-27:7, 50:5-7 (“Registration Transactions: The protocol used between repositories to establish a trusted session”). As they explain:

A usage transaction is carried out in a session between repositories. For usage transactions involving more than one repository . . . a registration transaction is performed. . . The goal of the registration

transaction is to establish a secure channel between two repositories who know each others identities.

Ex. 1001 (956) at 26:34-41.

B160. A person of ordinary skill in the art would understand that the registration message is simply a standard part of the process used to initiate a secure session, which includes a certificate exchange. *See* Ex. 1017 (Denning) at 178-179.

B161. The term “**registration message**” should be construed to encompass the certificate exchange process during the initiation of a session.

13. '859 patent: means for storing the content

B162. Claims 3, 31, and 60 of the '859 patent specify that the claimed repository comprises “**means for storing the content.**” Claim 62 specifies that means for storing content further comprises “means for storing content after rendering.”

B163. The Stefik patents shows that a repository contains a “storage system 1207,” which can comprise different storage media such as solid state storage or an optical disk. Ex. 1001 (859) at 13:37-47, 13:20-22, Fig. 12. As they state, “the content storage will store the associated content.” *Id.* at 13:38-41. A repository also can store content in offline backup files stored on a “magneto-optical storage system or magnetic tape.” *Id.* at 37:12-16.

B164. In IPR2013-00137, the Board construed the phrase “means for storing the content” as “cover[ing] content storage, such as an optical disk, a magneto-optical storage system, a magnetic tape, or a solid state storage, and equivalents thereof.” IPR2013-00137, Paper 17 at 15-16 (July 1, 2013).

B165. The Board’s construction is consistent with the specification. Therefore, I have used the Board’s construction of the phrase “**means for storing the content**” in my analysis.

14. ’859 patent: means for communication with a master repository

B166. Claim 24 of the ’859 patent specifies that the rendering system of claim 1 comprises “means for communicating with a master repository for obtaining an identification certificate for the repository.”

B167. The Stefik patents show that a repository contains a “processing means 1200” and an “external interface means 1206.” Ex. 1001 (859) at 13:20-26, 13:37-38. The processing means controls the operation of the repository, including controller functions and executing transaction functions. *Id.* at 13:20-26. The processing means utilizes the external interface means to connect to and communicate with other repositories. *Id.* at 13:52-53. The external interface can be various standard hardware components and can allow the repository to communicate over a network. *Id.* at 13:52-59.

B168. In IPR2013-00137, the Board construed the phrase “means for communicating” as “an external interface that provides for a signal connection with another device and equivalents thereof.” Paper 17 at 16.

B169. The Board’s construction is consistent with the specification. Therefore, I have used the Board’s construction of the phrase “**means for communicating**” in my analysis.

15. ’576 patent: means plus function elements

B170. Claim 1 of the ’576 patent contains seven means elements, six of which specify functionality of the claimed repository.

B171. In its decision on institution in IPR2013-00139, for each means element, the Board identified the corresponding algorithm in the specification. IPR2013-00139, Paper 15 (“ZTE 576 Inst. Decision”) at 20-26. In my analysis I have used the constructions found by the Board. For convenience, I reproduce those constructions below.

B172. Claim 1 recites that the apparatus of claim 1 comprises “**means for requesting use of the digital content stored in the storage.**” To process that request, claim 1 recites that the repository includes “**means for processing a request from the means for requesting.**” *See* ZTE 576 Inst. Decision at 20-21.

B173. The following algorithm corresponds to these elements:

Processing begins when a requester repository has sent a message to initiate a request. (Ex. [1009], 36:35-40, 37:9-13.) First, the server

repository checks the compatibility of the requester repository and the validity of the requester's identification. (Ex. [1009], 36:41-44, 37:14-17.) Second, the requester and server repositories perform the common opening transaction steps of determining whether authorization is needed, and whether the requested transaction is permitted given the usage rights. (Ex. [1009], 36:45-46, 37:18-19, 30:59-31:49.) Third, requester and server repositories perform a transmission protocol to read and write blocks of data, and then the requester repository renders the digital work. (Ex. [1009], 36:47- 50, 37:20-22.) Fourth, the contents are removed from the rendering device and the requester repository. (Ex. [1009], 36:51-52, 37:23-24.) Finally, the requester and server repositories perform the common closing transaction steps of updating the usage rights and billing information. (Ex. [1009], 36:53-54, 37:25-26, 32:20-31.)

ZTE 576 Inst. Decision at 21-22.

B174. Claim 1 recites that the repository includes “means for checking whether the request is for a permitted rendering of the digital content in accordance with rights specified in the apparatus.” The '576 patent states that the print and play transactions both use the common opening transaction steps to determine whether the requested transaction is permitted. (Ex. 1009 at 36:45-46, 37:18-19, 30:59-31:49.). The following algorithm corresponds to this element:

First, the requester repository determines whether an authorization certificate or a digital ticket is needed. (Ex. [1009], 31:3-9.) Second, the server repository generates a transaction identifier. (Ex. [1009],

31:10-13.) Third, the server repository determines whether the right is granted and whether time, security, and access based conditions are satisfied. (Ex. [1009], 31:13-33.) Finally, the server repository determines whether there are sufficient copies of the work to distribute. (Ex. [1009], 31:34-49.)

ZTE 576 Inst. Decision at 22-23; *see also* Ex. 1009 (576) at 36:45-46, 37:18-19, 30:59-31:49.

B175. Claim 1 recites that the repository includes “means for checking whether the request is for a permitted rendering of the digital content in accordance with rights specified in the apparatus.” The Stefik patents state that the print and play transactions both use the common opening transaction steps to determine whether the requested transaction is permitted. (Ex. 1009 at 36:45-46, 37:18-19, 30:59-31:49.). The following algorithm corresponds to this element:

First, transaction information is provided to the server repository by the requester repository. (Ex. [1009], 33:3-8.) Second, the server repository transmits a block of data to the requester repository and waits for an acknowledgement provided by the requester when the block of data has been received. (Ex. [1009], 33:9-15.) Unless a communications failure terminates the transaction, that process repeats until there are no more blocks to transmit. (Ex. [1009], 33:16-38, 33:46-49.) Finally, the requester repository commits the transaction and sends an acknowledgement to the server repository. (Ex. [1009], 33:39-45.)

ZTE 576 Inst. Decision at 23; *see also* Ex. 1009 (576) at 36:46-49, 37:20-22.

B176. Claim 1 recites that the repository includes “**means for authorizing the repository for making the digital content available for rendering.**” The following algorithm corresponds to this element:

First, a communication channel is set up between the server and the remote repository. (Ex. [1009], 41:52-54.) Second, the server repository performs a registration process with the remote repository. (Ex. [1009], 41:55-57.) Third, the server repository invokes a “play” transaction to acquire the authorization object. (Ex. [1009], 41:58-64.) Finally, the server repository performs tests on the authorization object or executes a script before signaling that authorization has been granted for rendering content. (Ex. [1009], 41:65-42:16.)

ZTE 576 Inst. Decision at 24; *see also* Ex. 1009 (576) at 36:45-46, 37:18-19, 31:6-9, 41:31-42:16.

B177. Claim 1 recites that the repository includes “means for making a [request] for an authorization object required to be included within the repository for the apparatus to render the digital content.” The following algorithm corresponds to this element:

First, a communications channel is set up between the server repository and the remote repository. (Ex. 1001, 41:52-54.) Second, the server repository performs a registration process with the remote repository. (Ex. 1001, 41:55-57.) Third, the server repository invokes a “play” transaction to request the authorization object. (Ex. 1001, 41:58-64.)

ZTE 576 Inst. Decision at 25; *see also* Ex. 1009 (576) at 31:6-9, 41:31-42:16.

B178. Claim 1 recites that the repository includes “**means for receiving the authorization object when it is determined that the request should be granted.**” The following algorithm corresponds to this element:

[T]he server repository transmits a block of data to the requester repository and waits for an acknowledgement, which the requester provides when the block of data has been completely received. (Ex. [1009], 33:9- 15.) Unless a communications failure terminates the transaction, that process repeats until there are no more blocks to transmit. (Ex. [1009], 33:16-38, 33:46-49.) Finally, the requester repository sends a completion acknowledgement to the server repository. (Ex. [1009], 33:39-45.) In that regard, the specification states (Ex. [1009], 33:45-48): “The key property is that both the server and the requester cancel a transaction if it is interrupted before all of the data blocks are delivered, and commits to it if all of the data blocks have been delivered.”

ZTE 576 Inst. Decision at 26.

D. Prior Art References

B179. I understand that a U.S. patent is a formally published document, and that I may rely on the dates in the patent as to when the patent was filed and when it was granted.

B180. I understand that for printed publications, questions can arise whether it was publicly disseminated and when that occurred.

B181. I believe based on my personal experiences that each of the publications I refer to below was publicly disseminated without any restrictions before November of 1994, and often a year or more before that date.

1. Exhibit 1016 – ABYSS

B182. *ABYSS: A Trusted Architecture for Software Protection* (“ABYSS”), Proceedings of the IEEE Symposium on Security and Privacy (April 27-29, 1987, Oakland, California), 38-51 (1987) (ISBN: 0-8186-0771-8) is an article written by Steve R. White and Liam Comerford that was published in 1987. Ex. 1016.

B183. ABYSS indicates that it was published in the Proceedings of the IEEE Computer Society Conference on Security and Privacy, pages 38-51, 1987, in 1987.

B184. ABYSS also indicates it was presented at the IEEE Computer Society Conference on Security and Privacy, which took place on April 27-29, 1987.

B185. I understand that because ABYSS was published in 1987, several years before the November 1994 effective filing date of the Stefik patents, it is prior art for the claims of the Stefik patents under 35 U.S.C. § 102(b).

2. Exhibit 1017 – Denning

B186. *Cryptography and Data Security* (“Denning”) is a textbook written by Dorothy Denning. Ex. 1017. I have used this text when teaching my cryptography course.

B187. I understand that Denning was published in 1983.

B188. I understand that because Denning was published in 1983, over a decade before the November 1994 effective filing date of the Stefik patents, it is prior art for the claims of the Stefik patents under 35 U.S.C. § 102(b).

3. Exhibit 1057 – Dyad

B189. *Dyad: A System for Using Physically Secure Coprocessors*, Journal of the Interactive Multimedia Association: Intellectual Property Project Proceedings, vol. 1., issue 1, 121-152 (January 1994). (“Dyad”) (Ex. 1057) is an article written by J.D. Tygar and Bennett Lee. Ex. 1057.

B190. Dyad is included in the Proceedings of Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment Conference (the “IMA” conference). The proceedings were published in the Journal of the Interactive Multimedia Association Intellectual Property Project in January 1994.

B191. I understand the Dyad paper was presented at the IMA conference in April 1993.

B192. I understand that Dyad also was published as a research paper at Carnegie Mellon University on May 4, 1991.

B193. I understand that because Dyad was published not later than January 1994, ten months before the November 1994 effective filing date of the Stefik patents, it is prior art for the claims of the Stefik patents under 35 U.S.C. § 102(a).

4. Exhibit 1058 – Linn

B194. Copyright and Information Services in the Context of the National Research and Education Network, Journal of the Interactive Multimedia Association: Intellectual Property Project Proceedings, vol. 1., issue 1, 9-20 (January 1994) (“Linn”) is an article written by Jerry Linn. Ex. 1058.

B195. Linn is a printed publication that was included in the Proceedings of Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment Conference, which was published in the Journal of the Interactive Multimedia Association Intellectual Property Project no later than January 1994.

B196. I understand the article was presented at that conference in April 1993.

B197. I understand that because Linn was published in January 1994, ten months before the November 1994 effective filing date of the Stefik patents, it is prior art for the claims of the Stefik patents under 35 U.S.C. § 102(a).

5. Exhibit 1059 – Kahin

B198. *The Strategic Environment for Protecting Multimedia*, Journal of the Interactive Multimedia Association: Intellectual Property Project Proceedings, *vol. 1, issue 1, 1-8 (January 1994)* (“Kahin”) is an article written by Brian Kahin. Ex. 1059.

B199. Kahin describes the conference at which the Dyad and Linn papers were presented. Kahin states:

On April 2 and 3, 1993, four organizations involved in networking and multimedia issues sponsored a two-day workshop at Harvard's John F. Kennedy School of Government to address the problem. These organizations - the Coalition for Networked Information, the Interactive Multimedia Association, the MIT Program on Digital Open High Resolution Systems, and the Information Infrastructure Project in the Kennedy School's Science, Technology and Public Policy Program- represented a set of different perspectives on what all saw as a broad common problem. The workshop was designed to:

- map the territory between secure systems and the need for practical, user-friendly systems for marketing information resources and services;
- survey the technological landscape, evaluate the potential benefits and risks of different mechanisms, define a research agenda, and frame related implementation and policy issues;
- consider how and where within the overall infrastructure different technologies are best implemented; and

- present and analyze models for explaining protection systems and strategies.

Speakers were invited to address these issues along with the potential roles of particular technologies and mechanisms: billing servers; type-ofservice identifiers; header descriptors; labeling and tagging; fingerprinting; digital signatures; contracting mechanisms; EDI (electronic data interchange); copy protection; serial copy management; authentication servers; software envelopes; encryption; display-only systems; concurrent use limitations; and structured charging.

B200. Kahin also reports that Mr. Kahin was affiliated with the J.F. Kennedy School of Government, and was the General Counsel for the Interactive Multimedia Association.

6. Exhibit 1018 – Kahn

B201. Exhibit 1018 (Kahn) is a copy of United States Patent No. 6,135,646 to Robert E. Kahn and David K. Ely.

B202. I understand that because Kahn was a continuation of an application filed on October 22, 1993, over one year before the November 1994 effective filing date of the Stefik patents, it is prior art for the claims of the Stefik patent under 35 U.S.C. § 102(e).

7. Exhibit 1020 - Rosen

B203. Exhibit 1020 (Rosen) is a copy of United States Patent No. 5,557,518 to Sholom Rosen.

B204. I understand that because Rosen was filed on April 28, 1994, almost six months before the November 1994 effective filing date of the Stefik patents, it is prior art for the claims of the Stefik patent under 35 U.S.C. § 102(e).

8. Exhibit 1021 - Hartrick

B205. Exhibit 1021 is a copy of European Patent No. EP 0 567 800 to Hartrick (“Hartrick”).

B206. I understand that because Hartrick was published on November 3, 1993, over one year before the November 1994 effective filing date of the Stefik patents, it is prior art for the claims of the Stefik patent under 35 U.S.C. § 102(a).

9. Exhibit 1061 – ABYSS 1990

B207. *ABYSS: An Architecture for Software Protection*, IEEE Transactions on Software Engineering, Vol. 16., No. 6, 619-629 (June 1990) (“ABYSS 1990”) is a paper by Steve R. White and Liam Comerford.

B208. ABYSS 1990 is a later published version of the ABYSS paper (Ex. 1016) that largely overlaps with the content of that earlier paper. It is also authored by the same individuals (White and Comerford).

Patentability Analysis of the Claims of the Stefik Patents

VI. SECTION C: ABYSS (Exhibit 1016)

A. Overview of ABYSS

C1. The ABYSS article (Ex. 1016) describes a copy protection technique that can be implemented using features of the ABYSS architecture.

C2. The ABYSS architecture was developed by IBM in the 1980s. ABYSS stands for A Basic Yorktown Security System. ABYSS indicates that the ABYSS platform is “shown to be a general security base, in which many security applications may execute.” Ex. 1016 (ABYSS) at 38. .

C3. The described technique enables “the trusted execution of application software, and can be used as a uniform security service across the range of computing systems.” Ex. 1016 (ABYSS) at 39. As ABYSS explains:

The use of ABYSS discussed in this paper is oriented towards **solving the software protection problem**, especially in the lower end of the market. Both current and planned software distribution channels are supportable by the architecture, and the system is nearly transparent to legitimate users. . . . **Software vendors may use the system to obtain technical enforcement of virtually any terms and conditions of the sale of their software, including such things as rental software.**

Software may be transferred between systems, and backed up to guard against loss in case of failure. We discuss the problem of protecting software on these systems, and offer guidelines to its solution.

ABYSS is shown to be a general security base, in which many security applications may execute.

Ex. 1016 (ABYSS) at 38 (emphasis added).

C4. The ABYSS architecture provides a “general security base,” which means that the techniques for providing security to software distribution can be applied to other applications and to other content.

C5. In the ABYSS architecture, a “protected processor” is included in every computer system. Ex. 1016 (ABYSS) at 39. The protected processor is a tamper-resistant module that provides cryptographic security, and ensures that encryption keys remain hidden from a user. *Id.*

C6. ABYSS explains how, using the ABYSS framework, a software vendor can attach terms and conditions to a software product, and compliance with those terms and conditions will be enforced by the protected processor. Ex. 1016 (ABYSS) at 39 (“The conditions under which an application may execute are embodied in a logical object called a Right-To-Execute. **These conditions are enforced by the protected processor.**” (emphasis added)); *id.* at 38 (“Software vendors may use the system to obtain technical enforcement of virtually any terms and conditions of the sale of their software, including such things as rental software.”), 50; *id.* at 39 (“The ABYSS architecture provides the software vendor with tools to enforce the conditions under which the application may be used.

Software run under ABYSS executes exactly as it was written, and cannot be modified arbitrarily by the user.”); *id.* at 40 (“Rights-To-Execute are . . . used by the supervisor to control the entire range of actions that can be taken with respect to the application . . . For instance, the software vendor may choose not to allow the Right-To-Execute to be transferred to another protected processor once it is installed.”).

C7. The software vendor distributes the software in encrypted or partially encrypted form along with a separate file, called a “Right-To-Execute”, containing the decryption key, the terms and conditions of use, and other data. Ex. 1016 (ABYSS) at 39, 44-45.

C8. Protected software can be executed on a computer only if the associated Right-To-Execute has been installed into the computer’s protected processor. Ex. 1016 (ABYSS) at 39.

C9. When a user tries to execute a software application, the protected processor will check to see if it has the corresponding Right-To-Execute. Ex. 1016 (ABYSS) at 44-45. If it does, it will verify that certain terms and conditions specified in the Right-To-Execute have been satisfied before allowing execution of the software. Ex. 1016 (ABYSS) at 44-45 (“The protected processor obtains reference information, which allows it to locate the Right-To-Execute to be

used. . . . Once found, the protected processor checks that this Right-To-Execute may be used to execute application programs.”).

C10. If the Right-To-Execute exists and the terms and conditions are satisfied, the protected processor will use information in the Right-To-Execute to decrypt the software and execute it. Ex. 1016 (ABYSS) at 45.

1. Basic Architecture of ABYSS

C11. The basic architecture of the ABYSS protected processor system is shown in Figure 1, which I have reproduced below:

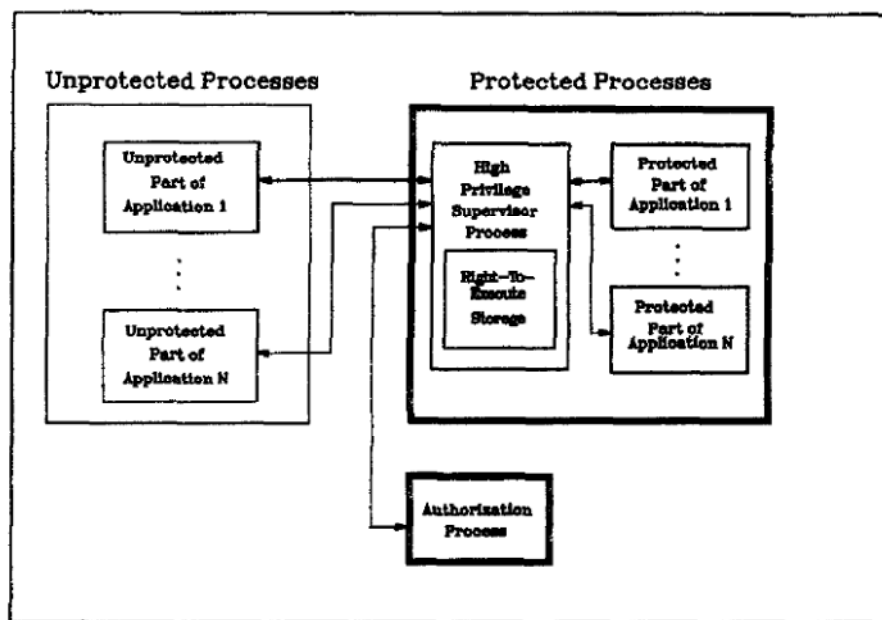


Figure 1. The Architecture of a Protected Processor System

Ex. 1016 (ABYSS) at 39.

C12. ABYSS describes Figure 1 as follows:

The architecture of the system presented here is shown in Figure 1.

Applications are partitioned into processes which are protected, and

processes which are not. Protected application processes are executed within **a secure computing environment called a protected processor**. The conditions under which an application may execute are embodied in a logical object called a Right-To-Execute. **These conditions are enforced by the protected processor**. The movement of Rights-To-Execute into and out of protected processors may require authorization from externally-supplied authorization processes.

Ex. 1016 (ABYSS) at 39 (emphasis added).

C13. As the passage explains, protected applications will be executed within the protected processor (the secure computing environment) and any conditions on execution will be “enforced by the protected processor.” *Id.*

C14. The protected processor can be used to provide security services because the processor is “a logically, physically, and procedurally secure unit.”

Ex. 1016 (ABYSS) at 39. As ABYSS explains:

It is **logically secure**, in that an application cannot directly access the supervisor process, or the protected part of any other application, to violate their protection. It is **physically secure** (which is indicated by the heavy box in Figure 1), in that it is contained in a tamper-resistant package It is **procedurally secure** in that the services which move information, and Rights-To-Execute in particular, into and out of the protected processor cannot be used to subvert the protection.

Ex. 1016 (ABYSS) at 39 (emphasis added).

C15. ABYSS also explains a similar point at page 50:

ABYSS is an architecture that enables the trusted execution of applications on protected processors, through its use of **logical, physical, and procedural security**. Software is separated from its Right-To-Execute, and strong cryptographic methods are used to manage both. Secure cryptographic channels can be used to move both software and Rights-To-Execute between protected processors.

C16. The ABYSS protected processor is physically secure because it is covered by a special coating, and if the coating is physically disturbed or damaged, the processor will erase the memory of the unit. Ex. 1016 (ABYSS) at 39.

C17. ABYSS explains that the protected processor can be the only processor in the system, or it can be a coprocessor that is added to a system containing another processor. Ex. 1016 (ABYSS) at 39.

2. The ABYSS Supervisor Process

C18. As shown in Figure 1 (¶ C11, above), the protected processor runs a “supervisor process” that ensures the logical and procedural security of the system. The supervisor process provides logical security as follows:

It manages the system’s communications resources, to ensure that messages are routed correctly between the protected and unprotected parts of applications. It executes at a higher privilege level than the application processes, and restricts them to isolated protection domains This isolation of application processes from each other, and from their unprotected parts, protects an application process from

attacks originating in other application processes, or in the unprotected parts of the computer system.

Ex. 1016 (ABYSS) at 40.

C19. ABYSS explains that a supervisor process must guarantee the procedural security of the system by ensuring four things:

In order to guarantee the procedural security of the system, each [supervisor] service must ensure four things. It must ensure that no cryptographic keys are exposed in plaintext, that protected software is not exposed in plaintext, that protected software only executes under the terms and conditions chosen by the software vendor, and that the protected software has not been modified.

Ex. 1016 (ABYSS) at 43.

C20. ABYSS explains that a manufacturer of a protected processor can customize the supervisor process. As the passage on page 43 shows, ABYSS describes the characteristics that a supervisor process must possess. As long as a supervisor process has those characteristics, many different processes are possible.

C21. A supervisor process could be configured to carry out public-key operations. *Id.* at 43 (“This section describes the supervisor services which were introduced in the previous section. Each of these services can employ either a symmetric cryptosystem or a public key cryptosystem. . . . Each service could use [different techniques] . . . as authorization mechanisms. . . .”). This would involve writing some code (or including a commercially available library, *e.g.*, an RSA

encryption library) to configure the protected processor to perform the sequence of operations involved in performing a public key encryption or decryption sequence.

C22. As another example, a process could implement additional security checks on a Right-To-Execute. *Id.* at 43 (“This section describes the supervisor services which were introduced in the previous section. Each of these services can employ either a symmetric cryptosystem or a public key cryptosystem. . . . Each service could use [different techniques] . . . as authorization mechanisms. . . .”).

3. Supervisor Keys

C23. Each protected processor stores one or more “supervisor keys” that can be used to decrypt rights-to-execute and to establish communications with other protected processors. Each processor also contains at least one supervisor key associated with its manufacturer. *Id.* at 44 (“For simplicity at this point, consider the supervisor key S to be common to all protected processors made by a given manufacturer. Alternatives will be discussed later.”).

C24. ABYSS explains that a protected processor contains several supervisor keys for establishing different types of communications:

Each smart card also contains a supervisor key S_C which is common to all protected processors made by a given manufacturer. . . . Protected processors also contain common supervisor keys S_m and S_m' , which are used only for communicating with the hardware manufacturer. S_m is used only for sending information to the manufacturer, and S_m' is used only for receiving information from the manufacturer.

Ex. 1016 (ABYSS) at 45; *id.* at 48.

C25. A protected processor also can contain a “unique supervisor key” that allows “unique identification of any protected processor.” Ex. 1016 (ABYSS) at 49 (“Targeted distribution can be achieved with unique supervisor keys. The software vendor uses the unique supervisor key of the target system to encrypt the Right-To-Execute destined for that system.”).

C26. A protected processor also could contain additional supervisor keys, such as a key associating a processor with a particular company or organization. Ex. 1016 (ABYSS) at 48 (“Supervisor keys may be unique to a given protected processor, common to all processors made by a given manufacturer, **or anything in between.**” (emphasis added)) ; *id.* at 49 (“Some software packages, particularly expensive software systems, are distributed to only **a select group of users.** Often, the software vendor wants to have a close contractual relationship with the users, or wishes to maintain a detailed list of users for maintenance purposes. We call this targeted distribution, since it is to a select group of systems, known in advance by the software vendor.” (emphasis added)).

C27. Supervisor keys associating a processor with a company or group would be used to enable the “targeted distribution” of software to a particular group of users/processors. Ex. 1016 (ABYSS) at 49.

C28. ABYSS explains that, if two protected processors share a common supervisor key, the supervisor key can be used to negotiate a session key to create a secure communication session between the processors. Ex. 1016 (ABYSS) at 42 (“Where it is possible to establish a [direct communication] channel, protected processors possessing an encryption/decryption key in common can set up a cryptographically secure channel between them, mediated by a session key.”); *id.* at 47 (“Alice’s and Bob’s protected processors use a common supervisor key S_C , to establish a secure session with each other, mediated by the session key T .”), 49 (“Common supervisor keys also allow secure two-way communication between any two protected processors. This is advantageous for transferring and backing up Rights-To-Execute.”).

C29. Although ABYSS describes supervisor keys as symmetric keys, it also explains that “these services can employ either a symmetric cryptosystem or a public key cryptosystem.” Ex. 1016 (ABYSS) at 43.

C30. It would have been a simple design choice to implement the ABYSS system using asymmetric encryption and digital certificates. In November 1994, use of public key systems were well-known in the art, and implementing these standard techniques in the ABYSS system would have been a routine task for a person of ordinary skill in the art. I explain this in more detail in ¶¶ C76-C102, below. *See also* ¶¶ A134-A160, *above*.

C31. The secure inter-processor communication protocol makes possible complex functionality such as enabling a user to transfer or back-up software and a Right-To-Execute to another system and enabling software lending libraries. Ex. 1016 (ABYSS) at 38 (“Software may be transferred between systems, and backed up to guard against loss in case of failure.”); Ex. 1016 (ABYSS) at 49 (“This software lending library is much like a lending library for books, in which the books automatically reappear in the library when they are due.”).

4. Creating Protected Software

C32. To create protected software for the ABYSS system, a software vendor will fully or partially encrypt the application before distributing it. Ex. 1016 (ABYSS) at 43 (“Once the part P of the application to be protected is complete, it is encrypted under an application key A , chosen by the software vendor, to form $E_A(P)$. The application key may be unique to each application, or even to each copy of each application.”).

C33. A partially encrypted application is shown in Figure 1, and it has “protected parts” (which are encrypted) and “unprotected parts” (which are not).
See ¶ C11, above.

5. The “Right-to-Execute”

C34. A Right-To-Execute is a file separate from the protected software, and it contains various data that enable a protected processor to execute the associated software. ABYSS explains:

The software is separated from the right to execute it. Only authorized users of an application have a Right-To-Execute for that application. Rights-To-Execute are created by software vendors, and are used by the supervisor *to control the entire range of actions that can be taken with respect to the application.*

Ex. 1016 (ABYSS) at 40 (emphasis added).

C35. As I stated above in ¶¶ C6-C8, a protected software application can be used only if the associated Right-To-Execute has been installed into the protected processor. Ex. 1016 (ABYSS) at 39.

C36. After being installed, the Right-To-Execute is stored inside the protected processor to help ensure that it cannot be modified or stolen. Ex. 1016 (ABYSS) at 39 (the protected processor “also contains secure memory for storage of Rights-To-Execute.”), Figure 1 (¶ C11, *above*).

C37. ABYSS explains that a Right-To-Execute contains a decryption key for the protected software (which it calls an “application key”) and an identifier linking it to the protected software application. Ex. 1016 (ABYSS) at 40, 43.

C38. ABYSS explains that the decryption key can be unique to a particular copy of an application, or all copies of an application can be encrypted with the same key. Ex. 1016 (ABYSS) at 43 (“The application key may be **unique to** each application, or even to **each copy of each application.**”), 49 (“A software vendor can specify the conditions under which the Right-to-Execute can be used for an application, or **even for a particular copy of an application.**”).

C39. Where the decryption key is unique to each copy of an application, each copy would have its own unique Right-To-Execute that, among other things, contained the correct decryption key to enable the decryption of the protected elements of the application. In other words, the protected code elements of a particular copy of the application will be encrypted with a unique key and the unique key for that copy of the application will be stored in a unique Right-To-Execute.

C40. To uniquely associate the Right-To-Execute with the particular copy of the application, the identifying information or “reference information” in Right-To-Execute would be unique to the copy of the application. *See generally* Ex. 1016 (ABYSS) at 44.

C41. The Right-To-Execute contains a set of data that defines how the protected processor’s supervisor process can use the Right-To-Execute, and it can include other data that can be used to define how the application may be used.

A Right-To-Execute consists of:

- An encryption and/or decryption key for software packages. This is required to decrypt the application before execution.
- Identifying information about the Right-To-Execute. This can be used to aid the supervisor in searching for the applicable Right-To-Execute.
- **Information about how the Right-To-Execute may be used by supervisor software. This allows the software vendor to control what the supervisor may do with the Right-To-Execute. For instance, the software vendor may choose not to allow the Right-To-Execute to be transferred to another protected processor once it is installed.**
- **Information about how the supervisor may permit the Right-To-Execute to be used** by software decrypted under its key. The software vendor may wish to allow the application to change the information in the Right-To-Execute, for instance.
- **Information about how the supervisor may permit the Right-To-Execute to be used by non-supervisor software** which is not decrypted under its key. It may be useful to allow other applications to report certain information about the Right-To-Execute. For instance, a utility could summarize information about all Rights-To-Execute owned by a user.
- **Additional information, at the discretion of software decrypted under the above key.** As will be seen later, the application may store such things as an expiration date for its

Right-To-Execute, and be assured that the application will not execute after that date.

Ex. 1016 (ABYSS) at 40 (emphasis added).

C42. In the emphasized portions of the passage quoted above, ABYSS explains that the protected processor's supervisor process will enforce certain conditions specified in a Right-To-Execute. Ex. 1016 (ABYSS) at 40 (Right-To-Execute contains "[i]nformation about how the Right-To-Execute may be used by supervisor software").

C43. For example, the software vendor can specify that, once installed, a Right-To-Execute cannot be transferred to another protected processor, and this restriction is enforced by the supervisor process. Ex. 1016 (ABYSS) at 40 (" . . . For instance, **the software vendor may choose not to allow the Right-To-Execute to be transferred** to another protected processor once it is installed." (emphasis added)).

C44. The software vendor encrypts each Right-To-Execute with a supervisor key. Ex. 1016 (ABYSS) at 44. The Right-To-Execute can only be installed into a protected processor that possesses the corresponding supervisor key. As ABYSS explains:

A plaintext file RTE_A , containing all of the necessary information for the Right-To-Execute for the application, can be created by the

software vendor. It contains the application key A , and associated information about how it can be used.

This file is encrypted under a supervisor key S , which is possessed by the protected processor on which the application will be installed, to create $E_S(RTE_A)$. This encryption can be done by a protected processor, as a service to the software vendor. It can be done without revealing the supervisor key S , since the encryption can be done securely inside the protected processor. For simplicity at this point, consider the supervisor key S to be common to all protected processors made by a given manufacturer. Alternatives will be discussed later.

Ex. 1016 (ABYSS) at 44 (emphasis added); *see id.* at 48.

C45. Supervisor keys are used to identify different protected processors. When transferring or installing a Right-To-Execute, the protected processor's possession of an authorized supervisor key certifies that the particular protected processor can be trusted. Ex. 1016 (ABYSS) at 48 ("ABYSS allows each software vendor to distribute only to **protected processors made by manufacturers that they trust**. This is done by creating Rights-To-Execute which can be loaded only onto processors which have **the trusted manufacturer's supervisor key**." (emphasis added)). This allows software vendors to prevent software from being installed into a protected processor if that brand of protected processor has been corrupted, either physically (*e.g.*, the tamper-resistant packaging can be breached)

or in terms of software (*e.g.*, the supervisor process running the processor is not secure). *Id.*

6. Downloading Protected Software

C46. ABYSS shows that software can be distributed on a diskette or over a network. Ex. 1016 (ABYSS) at 42, 50. To download software over a network, a user's protected processor would send a request to the software vendor through the Internet or other network, and would obtain in response a copy of the software and a Right-To-Execute.

In environments in which there can be secure two-way communication between the software vendor and the user, Rights-To-Execute may be distributed on demand to individual workstations. Both software, and Rights-To-Execute can then be distributed on local or wide area networks, or by download from host systems to workstations.

Ex. 1016 (ABYSS) at 49.

C47. ABYSS explains the software can be distributed "on demand," meaning an end-user can request (*i.e.*, demand) the software and download it from the software vendor's protected processor. As ABYSS explains, both the software and the Right-To-Execute can be transferred through an encrypted connection with another protected processor. *Id.* at 50 ("Secure cryptographic channels can be used to move both software and Rights-To-Execute between protected processors.").

7. Authorization Processes (Installing Software)

C48. ABYSS explains that protected software cannot be installed unless the protected processor receives authorization from the software vendor. Ex. 1016 (ABYSS) at 40-41, 46; *id.* at 39 (“The movement of Rights-To-Execute into and out of protected processors may require authorization from externally-supplied authorization processes.”).

C49. The authorization is provided via an “authorization process,” and each software vendor can create its own process for obtaining authorization.

C50. ABYSS describes several different processes that a software vendor could use to provide authorization to a user. Ex. 1016 (ABYSS) at 44. As ABYSS explains:

There are several methods by which a user may install Rights-To-Execute on a protected processor. The point of each of these methods is to install the Right-To-Execute for a particular application into the secure, persistent memory of the protected processor, if and only if a valid authorization exists to do so. The use of protected processors in preparing and installing Rights-To-Execute creates a trusted path for distributing Rights-To-Execute.

Ex. 1016 (ABYSS) at 44.

C51. ABYSS explains authorization to install a Right-To-Execute can be obtained by communicating with a server over the Internet or a network. Ex. 1016 (ABYSS) at 44 (“There are many possible authorization processes, including ones

in which the user's protected processor must communicate with a processor capable of **cryptographic transactions**, in order to receive proper authorization.” (emphasis added)) *id.* at 43 (“*Each service could use* tokens, or *communication between protected processors, as authorization mechanisms.*”). Authorization can be received as part of a “cryptographic transaction.” *Id.* Examples of such a transaction would include the server providing the protected processor with a new supervisor key, a digital certificate, or a digitally signed message.

C52. The server providing authorization could be maintained by the software vendor, or it could be maintained by an entity authorized by the vendor, such as a lending library file server. *See id.* at 49.

C53. If the installation into the user's protected processor is authorized, the Right-To-Execute would be loaded into the protected processor. *See* Ex. 1016 (ABYSS) at 44 (“If the authorization was valid, the protected processor installs the Right-To-Execute RTE_A into its secure, persistent memory.”); *id.* at 50.

C54. ABYSS describes an alternative method of securely providing authorization called a “token process.” Ex. 1016 (ABYSS) at 41. Using this method, a token is created by the software vendor and sent to a user on a “token card” or a “smart card.” Ex. 1016 (ABYSS) at 40-41. The token “acts as a one-time-only authorization from the software vendor, to a protected processor which possesses the application key A.” Ex. 1016 (ABYSS) at 41. ABYSS explains that

tokens are designed to provide secure authorization to a user when it is not possible to establish inter-processor communications. Ex. 1016 (ABYSS) at 50 (“ABYSS does not require changes to current or planned software distribution methods.”).

8. Executing Software

C55. ABYSS explains that” Once a Right-To-Execute is installed on a protected processor, it may be used at any time to enable the execution of applications associated with it.” Ex. 1016 (ABYSS) at 44-45; *id.* at 43 (“From the user’s point of view, there is no change in the way in which the application is executed.”).

C56. When a user tries to execute a software application, the protected processor uses “reference information” in the application to identify the associated Right-To-Execute. Ex. 1016 (ABYSS) at 44.

C57. The protected processor will evaluate the terms and conditions of use to determine if the Right-To-Execute can be used to authorize execution of the software. *Id.* at 45. If so, it will load the Right-To-Execute into memory. *Id.* As ABYSS explains:

1. The protected processor obtains reference information, which allows it to locate the Right-To-Execute to be used.
2. **Once found, the protected processor checks that this Right-To-Execute may be used to execute application programs.** Assuming that it is, selected parts of the Right-To-Execute may be made

available for reading, and modification, by the application to be loaded.

Ex. 1016 (ABYSS) at 44-45 (emphasis added).

C58. After the Right-To-Execute is loaded into memory, the protected processor loads the encrypted software file. The protected processor decrypts the software file, and verifies its integrity by checking a message authentication code.

As ABYSS explains:

3. U is loaded into the unprotected memory area, and $E_A(P)$ is loaded into the protected memory area and decrypted. If the decryption of $E_A(P)$ yields a valid **message authentication or manipulation detection code**, execution of U and P is begun. If not, P is purged from protected memory.

Ex. 1016 (ABYSS) at 45 (emphasis added). ABYSS explains that message authentication codes help ensure the procedural security of the system by ensuring the software has not been modified. Ex. 1016 (ABYSS) at 43 (“to guarantee the procedural security of the system, each service must ensure . . . that the protected software has not been modified.”). If the MAC is valid, then the protected processor executes the software.

C59. ABYSS explains that it is important to prevent the execution of an application that has been modified, because “If the application could be modified in a known way, it could be instructed to reveal its entire plaintext content. Even if the application could be modified randomly, it is still possible that the random

modifications, executing in the protected processor, will cause the revelation of important information about the protected part.” Ex. 1016 (ABYSS) at 44.

C60. While ABYSS explains that application integrity should be verified before executing the software, a person of ordinary skill in the art would understand that an application’s integrity also should be checked before installing the application. First, it would be illogical to install an application that would never run – *i.e.*, because it was corrupted, doctored, or otherwise lacked integrity. Second, it was common practice to ensure that the user had possession of the entire software package before attempting to install it, which would be accomplished by checking a signature, hash, checksum, or other digest of the software. *See* ¶¶ A98-A109, *above*.

C61. When protected software is not within the protected processor, it is maintained in encrypted form. *Id.* at 50 (“Software is separated from its Right-To-Execute, and strong cryptographic methods are used to manage both.”).

C62. A user cannot bypass the security of the protected processor by modifying the encrypted software. As ABYSS explains, “The encrypted form of the protected part cannot be modified directly because of the cryptographic techniques used to detect if it has been tampered with.” Ex. 1016 (ABYSS) at 40.

C63. ABYSS explains the encryption and integrity checking processes as follows:

Two goals must be met by this encryption. ... Second, it must prevent the protected part of the application from being modified, even randomly, and executed. This is in support of the requirement that the protected processor execute applications exactly as they were written. ... The second goal can be met by **using message authentication or manipulation detection codes [] when encrypting** the protected part of the application, and **having the protected processor use them to authenticate the application before it is executed.**

Ex. 1016 (ABYSS) at 43-44 (emphasis added).

C64. In November 1994, many authentication codes were well-known in the art, and the selection of which type of code to use would have been a simple design choice. For example, one obvious alternative would be to use a digital signature. I explain this in more detail in ¶¶ C78-C89, below.

C65. A software vendor may choose to allow a software application they distribute to read the Right-To-Execute, which would enable the software application to itself enforce certain types of terms and conditions. Doing this would allow a software vendor to specify additional or novel terms and conditions that were not already supported by the protected processor. As ABYSS explains:

4. At any time during execution, including at the very start, the protected part of the application may be allowed to access (and perhaps modify) selected parts of its own Right-To-Execute. It may use this in conjunction with the real-time clock in the protected processor, for instance, to verify that it is not being executed after its

expiration date. **Should the protected part of the application find that it is being executed contrary to the terms and conditions specified in its Right-To-Execute, it may effectively terminate its own execution.**

Ex. 1016 (ABYSS) at 44-45 (emphasis added).

C66. This feature of ABYSS is one of the reasons why it was highly adaptable to meet a variety of needs and capabilities of software distributors – it enables them to include additional security and control mechanisms on top of the basic security capabilities of the ABYSS protected processor. See Ex. 1016 (ABYSS) at 39 (“The ABYSS architecture provides the software vendor with tools to enforce the conditions under which the application may be used.”). This also complements the supervisor process feature of ABYSS, which enables the developer to configure the ABYSS protected processor actions directly.

C67. During execution, the decrypted software and the decryption key are confined within the protected processor. This prevents the user from viewing the decrypted software or decryption key and bypassing the security. As ABYSS explains:

The protected part [of a software application] executes securely on the protected processor, in that it cannot be examined or modified by any party external to the protected processor. It cannot be examined externally while it is available for execution because of the logical and

physical security of the processor, which inhibit all external access to the plaintext of the protected part.

Ex. 1016 (ABYSS) at 40. The key for decrypting the protected software is managed by the supervisor process, and the process ensures that key is not exposed outside of the protected processor. Ex. 1016 (ABYSS) at 40 (“The supervisor process contains a cryptographic facility for managing encryption/decryption keys. This facility decrypts the protected parts of applications as they are loaded into the protected processor.”).

9. Purchasing, Borrowing, or Transferring Software

C68. As I explained above (¶¶ C46-C47), a protected processor on a workstation can connect to a protected processor owned by a software vendor and download a copy of a software application and Right-To-Execute. Ex. 1016 (ABYSS) at 50, 49 (“In environments in which there can be secure two-way communication between the software vendor and the user, Rights-To-Execute may be distributed on demand to individual workstations. Both software, and Rights-To-Execute can then be distributed on local or wide area networks, or by download from host systems to workstations.”).

C69. After an application has been downloaded and installed, the ABYSS system permits a user to transfer a Right-To-Execute to a different protected processor, but only if the software vendor has specified in the Right-To-Execute that such transfer is permitted. Ex. 1016 (ABYSS) at 40-41, 46; *id.* at 43 (“Once

installed, Rights-To-Execute are not limited to residing in one system forever. They may be transferred at will by the user, if this is permitted by the software vendor.”). The restriction on the transfer of software and its Right-To-Execute is enforced by the protected processor. A protected processor’s supervisor process will determine whether the transfer can be permitted by checking the terms and conditions of the Right-To-Execute. Ex. 1016 (ABYSS) at 49 (“For instance, the software vendor may choose not to allow a Right-To-Execute to be transferred to another system, once it is installed. This provides technical enforcement of the condition that execution only occur on a given system.”), 40 (“Rights-To-Execute are . . . used by the supervisor to control the entire range of actions that can be taken with respect to the application . . . For instance, the software vendor may choose not to allow the Right-To-Execute to be transferred to another protected processor once it is installed.”).

C70. Software and its Right-To-Execute can be transferred over a network where it is possible to establish a network connection between two processors. ABYSS explains that standard, well-known techniques may be used to create a secure connection:

Where it is possible to establish [a direct communication channel between two protected processors], protected processors possessing an encryption/decryption key in common can set up a cryptographically

secure channel between them, mediated by a session key. There are a number of standard ways to do this.

Ex. 1016 (ABYSS) at 42; *see* Ex. 1016 (ABYSS) at 43 (“The services are simplified for clarity. **An actual implementation would include communication handshaking, verification of the completion of each step of the service, and error recovery.**” (emphasis added)). Many processes for establishing secure communications were well-known in the art. *See, e.g.*, ¶¶ A134-A160, *above*; Ex. 1017 (Denning) at 170-179.

C71. The ABYSS system permits a user to back-up a Right-To-Execute to disk or on another protected processor. Ex. 1016 (ABYSS) at 43, 45-46.

10. Software Lending Library

C72. ABYSS explains that its copy protection method allows for the creation of software lending libraries. Ex. 1016 (ABYSS) at 49.

C73. In this example, ABYSS shows a file server on a local area network acting a software library for workstations on that network. A workstation on the network can borrow software by sending a request to the file server to obtain a copy of the application and a Right-To-Execute it. If the server determines that it has available rights to execute, it will provide the workstation with the software and rights. As ABYSS explains:

In an environment where a large number of workstations are connected to a network, such as a local area network with a file

server, significant economies can be achieved for widely used applications. A single copy of an application can be kept on a file server. **The protected processor associated with this server can store a Right-To-Execute which enables up to some fixed number of copies to execute, for instance. When a user requests the application, a single Right-To-Execute is also transferred to the workstation, and the server's count is decremented.** The transferred Right-To-Execute may require renewal every minute or so to continue to be valid. The workstation can request renewal of the Right-To-Execute from the server at any time, and thus continue to possess a valid Right-To-Execute as long as is necessary. If the workstation does not check back with the server, the server knows that the workstation's Right-To-Execute has expired, so the server can increment its count of Rights-To-Execute for that application. **This software lending library is much like a lending library for books,** in which the books automatically reappear in the library when they are due.

Ex. 1016 (ABYSS) at 49 (underline added).

C74. ABYSS also enables rental software, outside of the library scenario, by allowing a software vendor to specify an expiration date or a maximum number of uses for the software:

By storing *an expiration time and date*, or a maximum number of allowed uses of the application, it is possible to enforce the concept of *limited lifetime software*.

Rental software currently poses a significant problem to software vendors. If the rented application's Right-To-Execute contains a terminal date, and if the application checks this against the real-time clock whenever it is run, the rental agreement is enforced. After the end of the rental period, both the renting company and the software vendor are assured that the renter can no longer use the application.

Ex. 1016 (ABYSS) at 49.

C75. ABYSS explains this feature allows for the creation of demonstration software that is guaranteed to expire:

As an extension of this, application packages may be prepared for demonstration purposes, and be sold for a price lower than the normal application. A user can try the full application, not some subset of it, in the particular computing environment, and with the particular data, on which the application is to be used. If the demonstration package is guaranteed to expire after a fixed time, or after some number of uses, the software vendor can use demonstration software as a marketing aid, without losing sales of the application.

Ex. 1016 (ABYSS) at 49 (underline added).

B. Implementing the ABYSS System

1. Integrating Well-Known Public-Key Cryptography Techniques into the ABYSS System, Such as Those Described in Denning

C76. The ABYSS article was published in 1987. The discussions in the article focus more on “a symmetric cryptosystem such as DES, since it is the best

match to both the cost and performance requirements of today's lower-end computers." Ex. 1016 (ABYSS) at 43.

C77. ABYSS explains that the techniques it describes can be implemented using public-key cryptography:

This section describes the supervisor services which were introduced in the previous section. **Each of these services can employ either a symmetric cryptosystem or a public key cryptosystem.** In this discussion, they are oriented towards a symmetric cryptosystem such as DES, since it is the best match to both the cost and performance requirements of today's lower-end computers. . . . The services are simplified for clarity. **An actual implementation would include communication handshaking, verification of the completion of each step of the service, and error recovery.**

Ex. 1016 (ABYSS) at 43 (emphasis added).

C78. The ABYSS system was designed to have certain functional and security properties. As is typical in engineering design, these properties could have been implemented in various ways. In particular, the key management aspects of the system could have been designed using either conventional cryptography or public-key cryptography. In addition, the software integrity and authentication verification processes could have been designed to use public-key cryptography instead.

C79. In making these architectural and implementation choices, engineers must consider tradeoffs among the choices. For example, key management with public-key cryptography is simpler than with conventional cryptography, but public-key cryptography operations are typically slower to compute than are conventional cryptography operations. As another example, public-key cryptography requires some type of mechanism to distribute public-keys assuredly, such as a public-key certificate authority, which adds complexity, but it increases the flexibility of key distribution. In 1994, a person of ordinary skill in the art would have been cognizant of these choices and tradeoffs.

C80. Several components of the ABYSS system present themselves as obvious choices to implement using public-key techniques. These include: (i) the symmetric supervisor keys possessed by each processor, (ii) authentication techniques when initiating sessions, (iii) the “Message Authentication Code” (MAC) used to protect the integrity of software, and the authorization processes used to obtain authorization to install a Right-To-Execute. Each of these components were commonly implemented using public-key techniques, and there were known and predictable benefits to using those techniques.

C81. In 1994, a person of ordinary skill in the art would have known how to implement the ABYSS system using public-key pairs for supervisor keys, using digital certificates and certificate authorities to distribute and authenticate keys,

and using digital certificates to initiate secure communications. For example, such techniques are taught in *Cryptography and Data Security*, a well-known cryptography textbook written by Dorothy Denning (“Denning”). Ex. 1017 (Denning) at 11-14, 169-171, 178-179; *see* ¶¶ A134-A160, *above*. Denning was published in 1983, and it is cited in ABYSS. Ex. 1016 (ABYSS) at 51. Many other references discuss these techniques. *E.g.*, Ex. 1060 (Herzberg & Pinter) at 158 (also cited in ABYSS).

C82. Denning describes a standard technique for initiating a secure communication link by exchanging certificates. Ex. 1017 (Denning) at 178-179. In this routine, the client *A* (the device initiating the connection) obtains and validates a digital certificate for a server *B* (the device being connected to). *A* then sends to *B* a message containing its certificate, a signed session key, and a timestamp, and it then encrypts the message with *B*’s public key (which is identified in the digital certificate). *B* validates *A*’s certificate and then sends a response that confirms the connection. *See id.* at 175. *B* also can send *A* a handshake message to confirm that *A* possesses the private key it claims to have. *Id.* Using this standard technique for initiating a secure connection using public keys, the digital certificate of the sender and receiver are authenticated, and if authentication fails, the session is not established.

a. **Benefits of Using Public Key and Certificate-Based Schemes in ABYSS Deployments**

C83. A person of ordinary skill in the art would understand that, in the configuration where public-private key pairs were used as supervisor keys, interprocessor communications would be initiated through a digital certificate exchange process, such as the one described in Denning, instead of by relying on processors having common symmetric keys.

C84. A protected processor receiving a connection request would authenticate the requester by validating the digital certificate, determining the certificate was issued by a trusted certificate authority, and verifying the requester possessed the private key it claims to have. Ex. 1017 (Denning) at 175, 178-79. It was routine in November 1994 to configure devices (*e.g.*, protected processors) to initiate secure communications by exchanging digital certificates.

C85. Similarly, software vendors could identify trusted hardware manufacturers using digital certificates. For example, instead of limiting the installation of its software to protected processors with particular symmetric supervisor keys, a software vendor would limit installation to processors that possessed a particular secret (private) key corresponding to a particular public key authenticated by a certificate issued by a trusted certificate authority. *See, e.g.*, Ex. 1048 (RFC 1422) at 11 (“A user may possess multiple certificates which may embody the same or different public components. For example, these certificates

might represent a current and a former organizational user identity and a residential user identity.”).

C86. In this configuration, when sending a Right-To-Execute to a protected processor, it would have been routine to encrypt the Right-To-Execute with the public key of the receiving processor. This would ensure that only the receiving processor could install it. When the Right-To-Execute is installed into the receiving processor, that processor can decrypt it to recover the clear text. If that protected processor later transferred the software and Right-To-Execute, it could use the same process (encrypt the Right-To-Execute with the public key of the receiving device). One of skill would have recognized the security benefits of this approach.

C87. It was also known to be useful to be able to dynamically identify processors or manufacturers that are no longer trusted. Certificate authorities provide a mechanism for doing that. Certificate revocation lists, which list certificates that have been revoked, were a well-known feature of certificate authority systems. *See, e.g.*, Ex. 1048 (RFC 1422) at 17, 22-24. Possession of a valid digital certificate would attest that the protected processor was trusted. Thus, a benefit of using certificates is that the certificate authorities can provide a revocation mechanism for identifying processors and manufacturers that are no longer trusted.

C88. In ABYSS, the main benefit of replacing conventional cryptography with public-key cryptography is to simplify the key-management aspects of the system. A disadvantage is that public-key operations are slower. However, when used for key management purposes (as opposed, for example, to bulk encryption), relatively few public-key operations would need to be performed. In 1994, a person of ordinary skill in the art would recognize that in balance there would be an overall benefit to using public-key cryptography.

C89. To the extent such techniques are not explicitly taught by ABYSS, substituting public-key encryption techniques into the ABYSS system would have been an obvious implementation of the ABYSS system, and a person of ordinary skill in the art would have recognized that in 1994. An engineer would be motivated to incorporate public-key cryptography into ABYSS to simplify the key-management aspects, as these techniques were well-known by 1994, and would have been able to readily make such a modification. *E.g.*, Ex. 1017 (Denning) at 12-16, 109, 170-179.

b. Developments After 1987 and Before 1994 Improved the Public Key Infrastructure

C90. Although ABYSS suggests that using symmetric encryption may have been more cost-effective in 1987 for the system the authors wanted to create, it most certainly does not say the public key encryption should not be used. Moreover, a person of ordinary skill in the art in 1994 would have known that

well-known advances in certificate authority standards since 1987 would have alleviated many of the concerns about implementing a public-key solution.

C91. After the ABYSS reference was published, two standards were released detailing protocols for establishing certificate authorities and distributing public-key information. As I explained above, the release of the X.509 standard in 1988 and the release of the RFC 1422 in 1993 made it easier to use public-key cryptography because those standards established protocols for distributing public keys and for using digital certificates as proof of authentication or identity. *See ¶¶ A134-A160, above.* The increasing adoption of the standards also lowered any costs associated with using public-key cryptography because the key distribution infrastructure already was in place.

C92. Moreover, Dyad (Ex. 1057) explicitly teaches that the combination of the ABYSS copy protection technique with public-key cryptography and digital certificates would improve the operation of the ABYSS systems:

When secure coprocessors are added to a system, however, we can quite easily protect executables from being copied and illegally utilized by attackers. . . . **Public key cryptography** may be used to encrypt the entire software package or a key for use with a private key system such as DES. **When a user pays for the use of a program, a digitally signed certificate of the public key used by his secure coprocessor is sent to the software vendor. This certificate is signed by a key management center verifying that a given public**

key corresponds to a secure coprocessor, and is prima facie evidence that the public key is valid. The corresponding private key is stored only within the NVM of the secure coprocessor; thus, only the secure coprocessor will have full access to the proprietary software.

A more primitive version of the copy protection application for secure coprocessors originally appeared in [63].

Ex. 1057 (Dyad) at 131-132. Reference “[63]” is the ABYSS article. *Id.* at 151 (“[63] Steve R. White and Liam Comerford. ABYSS: A trusted architecture for software protection. In Proceedings of the IEEE Computer Society Conference on Security and Privacy, pages 38-51, 1987.”). Tygar and Yee do not explicitly state what they mean by “primitive”; I interpret their intent to mean that ABYSS was implemented using conventional (not public-key) cryptography.

C93. In 1994, public-key cryptography had been incorporated into other copy protection schemes with clear benefits. For example, Kahn (Ex. 1018 (Kahn) at 10:23-39-48, Figs. 3 & 4) shows use of PEM (RFC 1422) and certificate authorities to provide authorization and authentication services. *See* ¶¶ A152-A160, *above*; *see* ¶¶ D22-D27, *below* (Kahn). Similarly, Rosen shows use of digital certificates and certificate authorities in conjunction with tamper-resistant hardware. *See* ¶¶ A152-A160, *above*; *see* ¶¶ E28-E37, *below* (Rosen).

C94. In 1994, it would have been a simple design choice for a person of ordinary skill in the art to have incorporated well-known certificate authority and

public-key cryptography techniques into the ABYSS architecture. In such a combination, public-key cryptography and certificate authorities would be simply performing the same function it had been known to perform, would have represented the combination of familiar elements according to known methods, and would yield no more than one would expect from such an arrangement.

c. Data Integrity and Authentication Functionality

C95. A person of ordinary skill would have recognized in 1994 that, in addition to using public-key cryptography to initiate communications between protected processors, public key cryptography could be used to implement the software integrity and authentication verification processes of the ABYSS system.

C96. To implement the *data integrity* security properties of the ABYSS architecture, a person could choose among several alternatives, including cryptographic checksums, hash functions, message authentication codes, and digital signatures of hashed values. In 1994, a person of ordinary skill in the art would have been familiar with these choices. Ex. 1017 (Denning) at 14-16 (PDF); Ex. 1057 (Dyad) at 129 (“Along with integrity, secure coprocessors offer privacy; this property allows the use of a wider class of cryptographic checksum functions. **There are many cryptographic checksum functions that might be used, including Rivest's MD5 [44] . . . [and] Jueneman's Message Authentication Code (MAC) code [26] . . .**” (emphasis added)).

C97. To implement the *sender authentication* security properties in ABYSS, an engineer could choose among several choices, including conventional cryptography, keyed message authentication codes, and digital signatures. In 1994, a person of ordinary skill in the art would have been familiar with these choices. Ex. 1017 (Denning) at 14-16 (PDF); Ex. 1057 (Dyad).

C98. ABYSS explains that the system ensures the integrity of software by checking a MAC before executing. A person of ordinary skill would have known that the MAC could be replaced with a digital signature and digital certificate, as it was well-known that using digital signatures to provide data integrity and sender authentication to certify software was advantageous. See ¶¶ A134-A145, *above* (Denning at 15-16).

C99. Whenever using digital signatures or public-key cryptography, it is natural also to use digital certificates to facilitate the distribution of authenticated public keys. Public keys are needed to encrypt messages and to authenticate digital signatures. In 1994, a person of ordinary skill in the art would have been familiar with these well-known techniques. Ex. 1017 (Denning) at 14-16 (PDF); Ex. 1057 (Dyad).

C100. The comments I made about integrity checking and authentication of digital data apply both to source code that is *protected content* and to source code that *operates the system that protects the content*. For simplicity, I will refer to

repository content as “software applications,” and I shall refer to software that operates the repository as “system software.”

C101. Concerning software applications (that is repository content), it is prudent to check the integrity and authenticity of the software both before installing it and before executing it. Ex. 1017 (Denning) at 15-16; Ex. 1033 (Davida) at 314-15. In 1994 a person of ordinary skill in the art would have been familiar with well-known choices for doing so, including cryptographic checksums, hash functions, message authentication codes, or a digital signature of the hash of the source code. It would have been obvious to use any of these techniques, including a digital signature of the hash of the source code, where ABYSS uses message authentication codes to validate software before executing it. Doing so would simplify the key management in the authentication and validation of software applications. Ex. 1016 (ABYSS) at 433-44, 44-45 (*see* ¶¶ C58-C64, *above*); Ex. 1057 (Dyad) at 129 (“There are many cryptographic checksum functions that might be used, including Rivest's MD5 [44] . . . [and] Jueneman's Message Authentication Code (MAC) code [26] . . .”).

C102. A person of ordinary skill in the art also would have found it obvious to authenticate and check the integrity of any system software (that operates the protected processor), such as updates to the system software, in a similar fashion, by validating a digital signature.

C103. ABYSS explains that the protected processor will authenticate software and verify its integrity before executing it. Ex. 1016 (ABYSS) at 433-44, 44-45 (*see* ¶¶ C58-C64, *above*). It would have been obvious to apply this technique to software that operates the protected processor as well. In addition, such an approach would have been obvious to implement in an ABYSS deployment because validating the integrity of operating system or kernel level software is explicitly taught by Dyad. Ex. 1057 (Dyad) at 138 (“[A] secure coprocessor verifies the system software (operating system kernel, system related user-level software) by checking the software’s signature against known values.”), 140 (“A core portion of the secure coprocessor software is code to manage keys. **Authentication, key management, fingerprints, and encryption crucially protect the integrity of the secure coprocessor software** and the secrecy of private data, including the secure coprocessor kernel itself.”); *id.* at 127-130.

d. **Using Public Key Techniques in Authorization Processes**

C104. One of skill would have also found it routine to implement public-key techniques into the ABYSS “authorization process.” As I explained above in C51, a person of ordinary skill reading ABYSS would understand that one common example of the “cryptographic transaction” described in ABYSS (Ex. 1016 at 44) was receiving a digitally signed authorization message, such as a receipt.

C105. When implementing public-key techniques into the system, one of ordinary skill would have been motivated to implement a system involving receiving an authorization message, such as a receipt that identifies the processor being authorized and an expiration date, that was digitally signed by the vendor. The protected processor could validate that the user received the digitally receipt before installing or even before executing the software.

2. The Software Lending Library

C106. As I mentioned above (§ C72-C73), ABYSS discloses a software lending library where a file server can distribute copies of an application and a Right-To-Execute to workstations on a network. To illustrate how the various features of the ABYSS system work together in action, I will explain how the lending library described in ABYSS would operate in practice.

C107. ABYSS describes the software lending library as follows:

In an environment where a large number of workstations are connected to a network, such as a local area network with a file server, significant economies can be achieved for widely used applications. A single copy of an application can be kept on a file server. **The protected processor associated with this server can store a Right-To-Execute which enables up to some fixed number of copies to execute, for instance. When a user requests the application, a single Right-To-Execute is also transferred to the workstation, and the server's count is decremented.** The transferred Right-To-Execute may require renewal every minute or so

to continue to be valid. The workstation can request renewal of the Right-To-Execute from the server at any time, and thus continue to possess a valid Right-To-Execute as long as is necessary. **If the workstation does not check back with the server, the server knows that the workstation's Right-To-Execute has expired, so the server can increment its count of Rights-To-Execute for that application.** This software lending library is much like a lending library for books, in which the books automatically reappear in the library when they are due.

Ex. 1016 (ABYSS) at 49 (emphasis added).

C108. The software lending library in ABYSS works because both the workstations and the file server contain protected processors, and the protected processors can be trusted to enforce the terms and conditions of use. A software vendor can rely on the protected processor in the file server to ensure the server does not lend or distribute too many copies or distribute copies to an unauthorized computer. Ex. 1016 (ABYSS) at 49. Similarly, the file server can rely on the protected processor in each workstation to ensure that a Right-To-Execute is disabled or returned to the server when the loan period expires. Ex. 1016 (ABYSS) at 49 (“If the workstation does not check back with the server, the server knows that the workstation's Right-To-Execute has expired”).

C109. I note that while ABYSS describes the loan period as approximately one minute long, it would have been a simple design choice to use a longer period

such as one hour or one day – in fact the actual period could be any value. Ex. 1016 (ABYSS) at 49 (“The transferred Right-To-Execute **may** require renewal every minute or so to continue to be valid.” (emphasis added)). It also would have been a simple design choice to have an unlimited loan period, allowing the file server to distribute a certain number of copies of software and rights-to-execute to workstations on the network.

a. **Distributing Software, Keys, and Certificates for the Library**

C110. To enable the software lending library, the file server must obtain a copy of the application and a Right-To-Execute from the software vendor. The file server can do this by initiating a secure connection with the software vendor. *Id.* at 49 (“In environments in which there can be secure two-way communication between the software vendor and the user, Rights-To-Execute may be distributed on demand to individual workstations. Both software, and Rights-To-Execute can then be distributed on local or wide area networks, or by download from host systems to workstations.”).

C111. The file server needs to be able to identify workstations that are part of the lending library. ABYSS explains that a vendor can distribute software to a “targeted” group of users by providing those users with a special supervisor key. Ex. 1016 (ABYSS) at 48, 49. ABYSS describes this configuration:

Some software packages, particularly expensive software systems, are distributed to only **a select group of users**. Often, the software vendor wants to have a close contractual relationship with the users, or wishes to maintain a detailed list of users for maintenance purposes. We call this targeted distribution, since it is to a select group of systems, known in advance by the software vendor.

Ex. 1016 (ABYSS) at 49 (emphasis added).

C112. While ABYSS does not explicitly describe using group supervisor keys to identify which protected processors are members of the lending library, a person of ordinary skill in the art would have found it logical and obvious to do so. One reason a vendor might want to create a special group supervisor key, would be to retain control over which protected processors can a particular software lending library and to prevent unauthorized users from joining.

C113. As I explained above, a person of ordinary skill in the art would have modified the supervisor keys to use public-private key pairs and digital certificates. *See ¶¶ C83, above.* Instead of issuing a “group” supervisor key to each member of library, each member would be issued a new digital certificate, issued by a particular authority, to identify it as a member. The software vendor could issue such certificates or it could delegate certificate issuing authority to another trusted entity, for example, an organization operating a software lending library file server. That entity could generate and issue local certificates to workstations, allowing a

local administrator to identify the computers that are part of group. The hierarchical structure of certificate authorities was a well-known feature of such systems. *See, e.g.*, Ex. 1048 (RFC 1422) at 20 (“Many of the CAs certified by PCAs are expected to represent organizations.”); *id.* at 11 (“... For example, these certificates might represent a current and a former organizational user identity and a residential user identity.”); Ex. 1020 (Rosen) at 10:40-11:38.

C114. As an alternative to digital certificates, members of the authorized group of users could be required to have a signed message from the software vendor or file server that identifies the protected processor as authorized. The signed message would serve as a proof-of-purchase or license file, identifying the processor as a member of a group. The signed message would be presented to the file server to show authorization.

C115. Use of signed messages for to establish authorization was a well-known technique by 1994. *E.g.*, Ex. 1018 (Kahn) at 23:39-45 (“The payment server signs the formatted message and sends it to the user's system. . . The user's system encrypts and stores (730) the received signed message. This account data will be submitted with any activity that may be billed.”); Ex. 1021 (Hartrick) at 17:7-26.

C116. Using either the digital certificate or digitally signed messages (or both) would provide obvious benefits to the system. For example, use of these

techniques simplifies the security protocols and reduces the risk if a particular processor is compromised or a key discovered. In addition, use of these public key techniques would obviate the need to transmit a group supervisor key over the network between two protected processors.

C117. To join the software lending library, a workstation would need to obtain the proper keys, digital certificates, or signed messages to show that it is authorized. ABYSS explains that protected processors obtain authorization using an “authorization process.” See ¶¶ C48-C54, *above*. ABYSS explains that authorization can be received via a “cryptographic transaction,” which would include receiving a group key, a digital certificate, or a digitally signed message. Ex. 1016 (ABYSS) at 44; *see also* ¶ C51, *above*. With public-key cryptography, authorization could take the form of a signed message authorizing the transfer. After a workstation obtained the necessary authorizations, it could participate in the lending library.

b. Borrowing a Copy of the Software from the Library

C118. As ABYSS explains, the user can borrow an application by transmitting a request to the file server. *Id.* at 49. Because rights-to-execute and software are only transferred between processors using secure channels, the file server’s protected processor and the workstation’s protected processor will attempt to initiate a secure channel. *Id.* at 40, 43, 49, 50 (“Secure cryptographic channels

can be used to move both software and Rights-To-Execute between protected processors.”).

C119. In this example, a secure channel would be initiated using the standard certificate exchange process, which I explained in detail above. *See* ¶¶ A148-A152 and ¶¶ C29, C51, *above*. To summarize, the workstation obtains the file server’s digital certificate, and then sends the file server a message containing its digital certificate, a session key, and timestamp, and encrypts the message with the file server’s public key. The file server validates the digital certificate, and uses a handshake message to verify the workstation possesses the corresponding private key. If everything verifies, the file server will allow the connection, and the session key will be used to encrypt subsequent communications. *See, e.g.,* Ex. 1016 (ABYSS) at 47 (“[P]rotected processors use a common supervisor key S_C , to establish a secure session with each other, mediated by the session key T .”).

C120. During the certificate exchange process, the file server both authenticates the workstation and determines whether it is authorized. Protected processors will only initiate secure communications channels with a trusted protected processor.

C121. ABYSS explains a protected processor determines whether another device is an authorized protected processor by verifying that the device has an authorized supervisor key, or digital certificate. Ex. 1016 (ABYSS) at 42, 43, 47,

48 (“ABYSS allows each software vendor to distribute only to **protected processors made by manufacturers that they trust**. This is done by creating Rights-To-Execute which can be loaded only onto processors which have **the trusted manufacturer’s supervisor key**.” (emphasis added)), 49 (“Common supervisor keys also allow secure two-way communication between any two protected processors.”). By verifying that the digital certificate was issued by a trusted authority, the file server determines the workstation is authorized.

C122. As I explained above (§ C114), alternatively, members of the lending library could be required to present the file server with a signed message from the software vendor to show authorization. After the session was established, the file server would verify that the workstation had a properly signed message. If so, the file server would determine the workstation was authorized, and would permit the workstation to request software. If the workstation’s protected processor could not provide the message, the file server would terminate the session. *See* § C115.

C123. Assuming the user was authorized, the file server would next check the Right-To-Execute associated with the requested application to determine whether any copies were available. If none were available, the file server would send the workstation an error message. If a copy was available, the file server’s protected processor would derive a Right-To-Execute for the workstation, set any expiration data, and then transmit the Right-To-Execute and a copy of the

application to the workstation's protected processor. *Id.* at 49. The file server would then decrement its count of available copies. *Id.*

C124. The requesting workstation would then receive a copy of the application and Right-To-Execute. These two items could be delivered together or separately – there is no functional consequence to either way. The workstation's protected processor would verify that it had authorization to install the Right-To-Execute (*e.g.*, by checking for the digitally signed receipt or by checking the other terms and conditions) into the protected processor. The protected processor would enforce the expiration period along with any other restrictions on use of the software. *Id.* at 49; *id.* at 39, 40, 43. If necessary, the workstation can periodically renew the Right-To-Execute; otherwise, when the Right-To-Execute expired, the protected processor would not permit the workstation to use the Right-To-Execute to run the application. *Id.* at 49.

C125. After the Right-To-Execute expired, the file server would increment its count of available copies. That copy then would be available for other workstations to borrow. *Id.* at 49.

C126. The protected processors in the file server and the workstation both would function as normal. Where a user requested to execute an installed application (*i.e.*, not borrowed), the protected processor would permit (or not) execution as normal. *Id.* at 40, 43, 44-45. The protected processors could be used

to obtain applications and Right-To-Execute directly from a software vendor. *Id.* at 49, 50. Or users could transfer a Right-To-Execute to another protected processor if that were permitted by the vendor. *Id.* at 43.

3. Implementation Variations

a. Usage Rights Language

C127. In ABYSS, the terms and conditions of use in a Right-To-Execute regulate the “entire range of actions” that can be taken with application, and specify how the application can be used (*e.g.*, executed or backed up) and distributed (*e.g.*, transferred or loaned). Ex. 1016 (ABYSS) at 39, 40. The terms and conditions also can include conditions such as an expiration date or a maximum number of uses.

C128. Although the ABYSS article does not provide details on how various actions are encoded within a Right-To-Execute, a person of ordinary skill would understand that there would have been several options for doing so.

C129. One option would have been to include static data fields corresponding to each right. Every Right-To-Execute would contain a block of data containing fields for each possible right and condition. For example, a series of bits in the file could be used to represent each right, with a “1” meaning the right was granted and a “0” meaning it was not. Each condition would have its own data field, and the condition would be present if data was contained in the

field (*e.g.*, a date and time appeared in the expiration date field as opposed to a null value).

C130. Another option would have been to have a dynamic set of data in the Right-To-Execute, where each a data field for a right or condition would be present only if that right or condition were included in the Right-To-Execute. In this configuration, every Right-To-Execute would contain a variably sized field for storing the rights and conditions. Each right and condition would be represented by a pair (or triple) of fields, with one data field used to identify the right and another field used to indicate whether any conditions are placed on the right. A third field could be included if the system needed to track data relating to the condition, *e.g.*, the number of copies currently lent out.

C131. Another option would be to include statements written in a programming language or structured document language such as SGML that can define how the software could be used. *See* Ex. 1062 (HTMLv2.0) at 2, 10-12 (“HTML as an Application of SGML . . . SGML is a system for defining structured document types and markup languages to represent instances of those document types[SGML].”); Ex. 1040 (Thinking in PostScript) at 1-6 (“PostScript as a programming language”).

C132. For example, Hartrick describes how specialized SGML tags can be used to specify the different actions that can be taken with respect to a file and to

specify additional fees that are charged for exercising certain rights. I describe these tags in more detail in ¶¶ E74-E98, below.

C133. A person of ordinary skill in the art would have been able to implement the ABYSS system using any of these approaches. Selecting among these techniques was a typical part of implementing a system in the context of the particular project and environment, and the selection would have been a standard engineering design choice.

b. Distributing Other Types of Digital Content

C134. While the ABYSS article is directed toward protecting software, its techniques can be applied to other media as well. As ABYSS explains:

We discuss the problem of protecting software on these systems, and offer guidelines to its solution. **ABYSS is shown to be a general security base, in which many security applications may execute.**

Ex. 1016 (ABYSS) at 38.

C135. ABYSS also explains that the ABYSS architecture can be used to provide security for other applications.

An application which executes in a protected processor does so in exactly the manner specified by the software vendor. Attackers cannot alter the terms and conditions under which it executes, or its actions when it is executing. **In security applications, the protected processor serves as a trustworthy agent of the supplier of computing services**, much as it does for the software vendor. **This**

makes protected processors an ideal base for many security applications. . . .

By reusing the resources of the protected processor for each of these tasks, the incremental cost of providing security for each of them can be reduced dramatically.

Ex. 1016 (ABYSS) at 50.

C136. It would have been obvious to adapt the ABYSS system to work with multimedia such as video and audio files. *See ¶¶ A101-A109, A115-A119, above* (IMA discussion – Kahin article). As I explained above, in 1994, developing techniques for securing all types of content was at the forefront of the industry. For example, authors of papers published in the IMA conference proceedings explained that Dyad and similar copy protection mechanism could be used to protect all types of digital content. For example, the Linn article states, “All mechanisms are applicable to any information distributed over a computer network” Ex. 1019 (Linn) at 17. I describe Linn in detail in ¶¶ D50-D77, *below*.

C137. Software and other types of content (*e.g.*, articles, photos, videos) are similar in two respects. First, each can be represented as a sequence of characters and organized into computer-readable files. It is a fundamental principle of digital computers that software is data that can be stored in computer memory. Ex. 1039 (The Stored Program Concept). Therefore, any system that protects and manages software files can also be used to protect and manage other digital files.

C138. Using the ABYSS system to protect other types of content would be an obvious and logical extension of ABYSS. The protection systems for these other types of content would work in the same manner as those used in ABYSS for protected software. The content (*e.g.*, a music file, a book, or a movie) would be distributed in encrypted form along with a Right-To-Execute that contained the decryption key and terms and conditions. When a user attempted to use the content by requesting an application to render it, the application would send a request to the protected processor to decrypt the content. If the requested use was permitted, the protected processor would decrypt the content and provide it to the application for rendering. This configuration would work exactly as one would expect it work.

C139. A person of ordinary skill in the art would have found it obvious to create an application program for rendering content and that used the security features of the ABYSS system to enforce use and distribution restrictions. Linn describes rendering software that enforces usage information associated with all types of digital content. I describe Linn article in detail below (*see* ¶¶ D50-D77). To leverage the ABYSS security features, content would be distributed in encrypted form, and decryption keys would be securely stored in the protected processor. When a user requested to use content, the rendering software would check any terms and conditions associated with the content, and it would allow

decryption and rendering only if authorized. The rendering software would only execute as the author intended because the ABYSS system ensures the integrity of the software before executing it.

c. **Deploying ABYSS and Hartrick to Distribute Softcopy Books**

C140. A person of ordinary skill would recognize that one type of content that could be distributed by ABYSS would have been the softcopy books described in Hartrick.

C141. It would have required only routine effort to make changes to the ABYSS system as it is described to enable it to distribute the softbooks using the techniques shown in Hartrick. Ex. 1021 (Hartrick) at 1:3-8, 5:19-38; see also C64-C67. As ABYSS explains, it is an architecture or framework designed to provide a range of security services and applications. Ex. 1016 (ABYSS) at 50.

C142. One simply way that the Hartrick scheme could be deployed using ABYSS would be to distribute softcopy books in encrypted form, provide the processors with a version of the book viewing program described in Hartrick, and store the royalty information file used by that book reading program in the Right-To-Execute. See Ex. 1021 (Hartrick) at 5:48-52. When a user tried to access a softbook using the bookreader program, the program would send a request to the protected processor requesting the processor to decrypt the book. The protected processor would access the Right-To-Execute, check the terms and conditions

(which would be defined using the security and royalty SGML tags described in Hartrick), and then if the requested action was permitted, the protected processor would provide the softcopy book reading program with access to the content of the softbook.

C143. To enable ABYSS to support the full range of functionality described in Hartrick, the Right-To-Execute could store security and royalty tags for individual parts of the softbook. As Hartrick explains, the royalty tags can be linked to individual chapters of a book or to the entire book, and a particular book can have both types of tags. Ex. 1021 at 6:23-28. Hartrick also shows how “coordinates” can be used to store the tags in a separate file from the content but link the tags back to the different chapters within the “softbook” file. Ex. 1021 at Figs. 4 & 10, 6:23-28, 13:44-54.

C144. These tags and coordinates could be integrated into the Right-To-Execute. ABYSS explains that a Right-to-Execute can be configured in a wide variety of ways to meet the needs of any particular implementation. Ex. 1016 at 49 (“*any terms and conditions that can be embodied in data . . . can be enforced*”), 48-49 (distribution alternatives). One of ordinary skill would recognize the Hartrick softbook tagging scheme could be readily implemented within the ABYSS Right-to-Execute. Ex. 1021 at Figs. 4, 7, 10.

C145. For example, a set of tags and corresponding coordinates could be stored in the Right-to-Execute for each softcopy book, including discrete sets of tags for the individual book chapters. Ex. 1021 at 5:48-52. The book would be accessed like the software in the ABYSS scheme – when access is requested, the ABYSS protected processors would locate the Right-to-Execute, obtain the relevant tag set for the part of the book being accessed, evaluate the information in them, particularly the royalty and security tags associated with different books or different chapters of each book, and determine if access can be granted.

C. ABYSS Shows “Usage Rights” and a “Repository”

1. ABYSS Shows “Usage Rights”

C146. As I explained above, a “usage right” is a statement in language (or data encoding the same) that contains a manner of use that identifies one manner of using or distributing a digital work and a conditions element that specifies any conditions associated with the specified right. These statements are permanently attached to a digital work and are enforced by a repository. ABYSS shows that the terms and conditions of use in the Right-To-Execute are “usage rights.”

C147. In ABYSS, the terms and conditions of use in a Right-To-Execute regulate how the associated application can be used and distributed. Ex. 1016 (ABYSS) at 39, 40 (Right-To-Execute is used to “control the entire range of actions that can be taken with respect to the application”).

C148. The terms and conditions of use allow a vendor to specify one more or actions that may be taken with the software application, such as whether it can be executed, transferred to a different computer, or lent to other computers. *Id.* at 40 (“the software vendor may choose not to allow the Right-To-Execute to be **transferred** to another protected processor . . .”), 44-45 (“[T]he protected processor checks that this Right-To-Execute may be used to **execute** application programs. . .”), 49 (“software **lending** library”), 38, 39, 40, 43, 50. Each of these actions is a particular manner of using or distributing the software. *See* Ex. 1001 (859) at 18:35-19:52 (Possible rights include “play,” “copy,” “transfer,” and “loan”). Consequently, each is a manner of use.

C149. ABYSS shows that the terms and conditions in a Right-To-Execute also contain a conditions element. ABYSS shows, for example, that a Right-To-Execute can “stor[e] an expiration time and date, or a maximum number of allowed uses of the application,” which makes possible “to enforce the concept of limited lifetime software. Ex. 1016 (ABYSS) at 49; *see also* Ex. 1001 (859) at 18:21-25 (“conditions are copy count, control, time, access and fee conditions”). A Right-To-Execute also can specify the maximum number of copies of an application can be lent out to other processors. Ex. 1016 (ABYSS) at 49.

C150. ABYSS shows that a Right-To-Execute is permanently attached to each copy of an application. Ex. 1016 (ABYSS) at 49 (“A software vendor can

specify the conditions . . . *even for a particular copy of an application.*”). Each copy of an application can be encrypted with its own key, and it will be attached to the Right-To-Execute using “identifying information.” *See* ¶¶ C38-C40, *above*.

C151. ABYSS shows that a Right-To-Execute can specify more than one right, *e.g.*, both the right to execute and the right to transfer. *Id.* at 40. Therefore, ABYSS shows “usage rights” within the meaning of the claims.

C152. As I explained above (¶¶ C127-C133), it would have been obvious to use specialized SGML tags, such as those described in Hartrick, to represent the terms and conditions of use in the Right-To-Execute. For example, different tags could be used to specify different actions, such as an “execute” tag, a “transfer” tag, and a “lend” tag. Similarly, it would have been obvious to place conditions on those rights by also specifying an “expire” tag or a “max use” tag. SGML is extensible, and it was designed to allow the creation of different libraries of tags to be used in different implementations. Changing the available set of tags was a routine engineering design choice. *See* Ex. 1062 (HTMLv2.0) at 2, 10-12 (“SGML is a system for defining structured document types and markup languages to represent instances of those document types[SGML].”).

2. ABYSS Shows Use of a “Repository”

C153. As I explained in Section B above, the Stefik patents define a “repository” as “a trusted system” that “maintains physical, communications and behavioral integrity,” and that “supports usage rights.”

C154. As I explain below, ABYSS in view of Denning shows that it would have been obvious to configure the protected processor to be a “repository” within the meaning of the claims.

C155. ABYSS explains that software is distributed in encrypted form, along with a Right-To-Execute that contains the terms and conditions of use of the software. *See* ¶¶ C6-C8, C32-C45, *above*. The protected processor will enforce compliance with a software application’s terms and conditions of use. *See* ¶¶ C8-C17, *above*; *see* Ex. 1016 (ABYSS) at 39, 38, 40, 50. Therefore, ABYSS shows that the protected processor supports usage rights.

C156. ABYSS describes a “trusted architecture,” and explains that systems containing protected processor are trusted systems. *See* ¶¶ C2-C17, *above*.

ABYSS explains that software and its Right-To-Execute are encrypted, and a protected processor will allow a user to decrypt and access the software only if the user is authorized. Ex. 1016 (ABYSS) at 38, 39, 50. Therefore, the protected processor is a trusted system.

C157. The protected processor is a tamper-resistant module that includes a processor and “secure memory for storage of Rights-To-Execute.” Ex. 1016

(ABYSS) at 39. The protected processor can be used to store data securely in any storage device on the computer system by encrypting the data before storage and then storing the decryption key within the protected processor. *See* ¶¶ C55-C67, *above*. Therefore, the protected processor possesses physical integrity. *See* ¶¶ B65-B66, *above* (“physical integrity” means “preventing access to information by a non-trusted system.”).

C158. ABYSS explains that protected processors can communicate with each other using a cryptographically secure channel only if the two processors “possess an encryption/decryption key in common.” *See* ¶¶ C28-C31, *above*; Ex. 1016 (ABYSS) at 42, 43, 50. ABYSS shows that both software and its Right-To-Execute can be transferred between protected processors via a secure channel. *See* ¶¶ C69-C71, *above*; Ex. 1016 (ABYSS) at 49, 50.

C159. ABYSS explains that it is presenting a simplified description of the process, and standard communications technologies should be incorporated into the system. *Id.* (“The services are simplified for clarity. An actual implementation would include communication handshaking, verification of the completion of each step of the service, and error recovery.”). These standard techniques, such as handshaking and using timestamps (nonces) to prevent replay attacks, are disclosed in Denning. *E.g.*, Ex. 1017 (Denning) at 170, 173-175, 178-179.

C160. ABYSS explains that the communications can be implemented using symmetric or public-key techniques. *Id.* at 43 (“Each of these services can employ either a symmetric cryptosystem or a public key cryptosystem. . . .”). While ABYSS predominantly describes using symmetric supervisor keys to identify processors and initiate secure communications, a person of ordinary skill in the art would have been able to implement the system using standard public-key techniques.

C161. For example, digital certificates could be used to identify and authenticate protected processors, and the certificates could be used to initiate secure communications. Ex. 1017 (Denning) at 170, 173-175, 178-179. To the extent not explicitly disclosed, a person of ordinary skill in the art would have found this configuration to be obvious.

C162. Therefore, it would have been obvious to configure the protected processor to possess communications integrity.

C163. ABYSS explains that, before executing software, a protected processor will authenticate the source of the software and validate that it has not been modified by validating “message authentication or manipulation detection codes” that are recovered by decrypting the protected software. Ex. 1016 (ABYSS) at 43-44; *see id.* at 44-45. A keyed message authentication code (MAC) can be formed by making a hash of the message (the software) and encrypting it

with a symmetric key (a supervisor key). *See ¶¶ A134-A139, above.* If the authentication code is valid, that certifies the application is from the software vendor and has not been modified.

C164. A person of ordinary skill in the art would have recognized that a digital signature would be a public-key variant of a MAC, and could be used as a substitute for the MAC. *See ¶¶ A137-A143, above.* It was well-known that a digital signature validates the source and integrity of a message, which is the function of a keyed MAC.

C165. In addition, a person of ordinary skill in the art would have recognized that it was advantageous to distribute digitally signed software. For example, this standard technique is described in Denning, which explicitly describes some of the benefits to distributing software with a digital signature and digital certificate to validate the software. Ex. 1017 (Denning) at 15-16, 109, 170. As Denning explains:

The system (preferably hardware) could refuse to execute any program in privileged mode that is not properly signed by a program verifier, making it impossible for someone to substitute a program that could run in privileged mode and wreak havoc in the system.

Ex. 1017 (Denning) at 15.

C166. Denning explains that it is beneficial to distribute all software with digital signatures and certificates, and it also teaches that it is especially beneficial

for software that affects the operation of a system (*e.g.*, operating system software).

C167. A person of ordinary skill in the art would have understood that a digital signature and digital certificate could be used instead of the MAC described in ABYSS. This is because the digital signature performs the same function as the MAC – it verifies the integrity of the application, and provides assurance it was not modified after it was signed. In such a combination, public-key cryptography and certificate authorities would be simply performing the same function it had been known to perform and would yield no more than one would expect from such an arrangement.

C168. ABYSS explains that a software vendor can limit distribution of its software to processors made by manufacturers the vendor trusts. Ex. 1016 (ABYSS) at 48. A vendor can specify that, for an application to be installed into a particular processor, the processor must contain a supervisor key issued by a trusted manufacturer. *See* ¶¶ C23-C28, *above*. With public-key cryptography, where each trusted processor would have a unique private key, possession of a valid digital certificate for the corresponding public key would permit the identification of processors manufactured by trusted vendors.

C169. Therefore, it would have been obvious to configure the protected processor to possess behavioral integrity.

C170. Thus, ABYSS in view of Denning shows that it would have been obvious to configure the protected processor to be a repository within the meaning of the claims.

D. ABYSS Would Have Made the '859 Patent Obvious

1. The Claims of the '859 Patent

C171. The independent claims of the '859 patent describe a “rendering system” that can access content over a network and render content in accordance with usage rights attached to that content. A “rendering device” is coupled to a “distributed repository,” and the distributed repository can request content from another repository and it enforces usage rights when content is rendered by the rendering device.

C172. ABYSS explains how a computer system containing a protected processor (“rendering system”) executes software in accordance with the software’s terms and conditions of use (“render[s] content in accordance with usage rights”).

C173. The protected processor is a “distributed repository” that has both a “requester” “server mode of operation. The protected processor can send requests to other protected processors to obtain a copy of software and a right to execute (“requester mode of operation”). See ¶¶ C51, C70, C110, C126, *above* (request from the software vendor); See ¶¶ C106-C126, *above* (request from a software

lending library). A protected processor receives requests from users to execute software and to borrow a copy of an application and a Right-To-Execute (“server mode of operation”). ¶¶ C42-C44; C65-C67; C146-C152.

a. Claim 1

C174. The processes I described above disclose all the limitation of claim 1 of the ’859 patent. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>’859 Patent: Claim 1</u>	<u>Relevant Concepts and Paragraphs</u>
1. A rendering system adapted for use in a distributed system for managing use of content, said rendering system being operative to rendering content in accordance with usage rights associated with the content, said rendering system comprising:	<p><u>“content”</u></p> <p>Software applications. See ¶¶ C6</p> <p><u>“distributed system for managing use of content”</u></p> <p>Content can be obtained from software vendors or a file server, and authorization can be obtained over a network. See ¶¶ C6; C48-C52; C127-C133</p> <p><u>“rendering system”</u></p> <p>Computer system containing a protected processor. See ¶¶ C23-C59</p> <p><u>“rendering content in accordance with usage rights”</u></p> <p>Protected processor executing software based on data in the Right-To-Execute. See ¶¶ C23-C59</p>

<p>[1.a] a rendering device configured to render the content; and</p>	<p><u>“rendering device”</u></p> <p>computer system with a protected processor. See ¶¶ C23-C59; C76-C126 (repository security)</p>
<p>[1.b] a distributed repository coupled to said rendering device and including a requester mode of operation and server mode of operation,</p>	<p><u>“distributed repository”</u></p> <p>Workstation with a protected processor. ¶¶ C23-C59; C153-C170</p>
<p>[1.c] wherein the server mode of operation is operative to enforce usage rights associated with the content and permit the rendering device to render the content in accordance with a manner of use specified by the usage rights,</p>	<p>Protected processor in a user’s workstation enforcing Rights-to-Execute. ¶¶ C5-C13, C42-C44; C146-C152</p> <p>Protected processor in a file server enforcing Rights-to-Execute in response to requests to borrow or purchase software. ¶¶ C5-C13, C42-C44; C72-C75, C146-C152</p>
<p>[1.d] the requester mode of operation is operative to request access to content from another distributed repository, and</p>	<p>Protected processor requesting a copy of an application and a Right-To-Execute from other protected processors. ¶¶ C46-C54</p>
<p>[1.e] said distributed repository is operative to receive a request to render the content and permit the content to be rendered only if a manner of use specified in the request corresponds to a manner of use specified in the usage rights.</p>	<p>Protected processor enforcing terms and conditions of use. ¶¶ C5-C13, C42-C44.</p>

b. Claims 29 and 58 of the '859 Patent

C175. Independent claims 29 and 58 of the '859 patent are identical to claim 1, except claim 29 specifies a method and claim 58 specifies software. ABYSS discloses systems, methods, and software for implementing the techniques described in the article. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'859 Patent: Claim 29</u>	<u>'859 Patent: Claim 58</u>	<u>Relevant Concepts and Paragraphs</u>
29. A rendering method adapted for use in a distributed system for managing use of content, and operative to render content in accordance with usage rights associated with the content, said method comprising:	58. A computer readable medium including one or more computer readable instructions embedded therein for use in a distributed system for managing use of content, and operative to render content in accordance with usage rights associated with the content, said computer readable instructions configured to cause one or more computer processors to perform the steps of:	See Claim 1. <u>“computer readable instructions”</u> supervisor process C14-C20; C41-C45
[29.a] configuring a rendering device to render the content;	[58.a] configuring a rendering device to render the content;	See Claim 1.
[29.b] configuring a distributed repository coupled to said rendering device to include a requester mode of operation and server	[58.b] configuring a distributed repository coupled to said rendering device to include a requester mode of operation and	See Claim 1.

mode of operation;	server mode of operation;	
[29.c] enforcing usage rights associated with the content and permitting the rendering device to render the content in accordance with a manner of use specified by the usage rights, when in the server mode of operation;	[58.c] enforcing usage rights associated with the content and permitting the rendering device to render the content in accordance with a manner of use specified by the usage rights, when in the server mode of operation;	<i>See Claim 1.</i>
[29.d] requesting access to content from another distributed repository, when in the requester mode of operation; and	[58.d] requesting access to content from another distributed repository, when in the requester mode of operation; and	<i>See Claim 1.</i>
[29.e] receiving by said distributed repository a request to render the content and permitting the content to be rendered only if a manner of use specified in the request corresponds to a manner of use specified in the usage rights.	[58.e] receiving by said distributed repository a request to render the content and permitting the content to be rendered only if a manner of use specified in the request corresponds to a manner of use specified in the usage rights.	<i>See Claim 1.</i>

C176. It is my opinion that ABYSS in view of Denning renders claims 1, 29, and 58 of the '859 patent obvious.

c. Claims 3, 31, 60 and Claims 5, 33, 62 of the '859 Patent

C177. Claims 3, 31, and 60 depend from claims 1, 29, and 58, respectively, and specify that “said repository comprises means for storing the content.”

C178. Claims 5, 33, and 62 depend from claims 3, 31, and 60, respectively, and specify that the “means for storing” comprises “means for storing content after rendering.”

C179. The protected processor is a tamper-resistant module that includes a processor and “secure memory for storage of Rights-To-Execute.” Ex. 1016 at 39, 44, 48, 50. The encrypted application can be stored in the secure memory (*id.* at 39), or in any storage device connected to the computer system so long as the protected processor has a Right-to-Execute containing the decryption key. *E.g.*, Ex. 1016 (ABYSS) at 39, 50, 40.

C180. ABYSS explains that when software is installed, it is copied into memory on the user’s computer. *Id.* at 42-43 (after receiving authorization to install “[t]he WonderCalc software may be copied, put onto a hard disk, etc.”); *see id.* at 38, 39, 40, 44-45, 49, 50.

C181. ABYSS shows that software is kept stored in memory after execution. Ex. 1016 at 39, 40, 42-43, 44-45, 49, 50.

C182. Therefore, ABYSS discloses a “*means for storing content*” both before and after rendering.

C183. It is my opinion that ABYSS in view of Denning renders claims 3, 31, and 60 and claims 5, 33, and 62 of the ’859 patent obvious.

d. Claims 8, 36, 65 and Claims 13, 40, 69 of the '859 Patent

C184. Claims 8, 36, and 65 depend from claims 5, 33, and 62, respectively, and specify that the content to be rendered comprises “video.”

C185. Claims 13, 40, and 69 depend from claims 1, 29, and 58, respectively, and specify that the “rendering device” comprises a “video system.”

C186. A person of ordinary skill would have understood that the ABYSS could be used to protect content such as video. *See* ¶¶ A101-A109, A115-A119, C134-C139, *above*.

C187. A computer that can play video content is a “video system.” ABYSS shows that a protected processor can be included in a “range of computing systems,” and a person of ordinary skill would have understood that the protected processor could be included in computer systems that could play videos. *See* ¶¶ C3, C134-C139, *above*.

C188. It is my opinion that ABYSS in view of Denning renders claims 8, 36, and 65 and claims 13, 40, and 69 of the '859 patent obvious.

e. Claims 15, 42 and 71 of the '859 Patent

C189. Claims 15, 42, and 71 depend from claims 1, 29, and 58, respectively, and specify that the “rendering device” comprises a “computer system” and the “repository” comprises “software executed on the computer system.”

C190. Each protected processor is installed into a computer system, and it can execute protected software. Ex. 1016 at 39. The supervisor process that controls the processor's operation is software. *Id.* Therefore, the repository comprises "software executed on the computer system."

C191. It is my opinion that ABYSS in view of Denning renders claims 15, 42, and 71 of the '859 patent obvious.

f. Claims 16, 43, 72 of the '859 Patent

C192. Claims 16, 43, and 72 depend from claims 1, 29, and 58, respectively, and specify that an "execution engine" is coupled to the repository and the repository will "permit said execution device to execute a computer program only in a manner specified by the usage rights."

C193. As I explained with respect to the independent claims, protected processors are installed in computer systems, execute software, and enforce terms and conditions of use.

C194. It is my opinion that ABYSS in view of Denning renders claims 16, 43, and 72 of the '859 patent obvious.

g. Claims 19, 46, 75 and Claims 20, 47, 76 of the '859 Patent

C195. Claims 19, 46, and 75 depend from claims 1, 29, and 58, respectively, and specify that the "manner of use is a manner of displaying."

C196. Claims 20, 47, and 76 depend from claims 1, 29, and 58, respectively, and specify that the “manner of use is a manner of playing.”

C197. ABYSS shows that “any terms and conditions that can be embodied in data in the protected processor, as a part of the Right-To-Execute, can be enforced.” A person of ordinary skill in the art would have found it obvious to include a way of displaying in the Right-To-Execute. For instance, for “demonstration” software intended to be distributed as a “marketing aid”, it would have been obvious to specify in the Right-To-Execute that program output be marked with a label, such as “Call to Order” or “Demo Version” to promote sales of the full version. Thus it would have been obvious to a person of ordinary skill to specify a “manner of displaying” in the Right-To-Execute. It is my opinion that ABYSS in view of Denning renders claims 19, 46, and 75 and claims 20, 47, and 76 of the ’859 patent obvious.

h. Claims 21, 48, 77 of the '859 Patent

C198. Claims 21, 48, and 77 depend from claims 1, 29, and 58, respectively, and specify that “the rendering device and the repository are integrated into a secure system having a secure boundary.”

C199. ABYSS explains that “[t]he protected processor is a *logically, physically, and procedurally secure unit*. . . It is physically secure (which is

indicated by the heavy box in Figure 1), in that it is contained in a tamper-resistant package. . .” Ex. 1016 at 39.

C200. The protected processor executes (“*renders*”) software and it provides integrity and enforces any Rights-To-Execute (“*repository*”). Therefore both the repository and rendering device are integrated into a secure system with a secure boundary.

C201. It is my opinion that ABYSS in view of Denning renders claims 21, 48, and 77 of the ’859 patent obvious.

i. Claims 24, 51, 81 of the '859 Patent

C202. Claims 24, 51, and 81 depend from claims 1, 29, and 58, respectively, and specify that the rendering system “comprises means for” or is “configured to” “communicat[ing] with a master repository for obtaining an identification certificate for the repository.”

C203. A person of ordinary skill implementing ABYSS would configure the system to use a public-key cryptosystem for the supervisor keys. In that configuration, a protected processor would obtain digital certificates by registering a public key with a certificate authority.

C204. For example, to join a lending library, a protected processor would register a public key with the vendor, and the vendor’s protected processor

(“*master repository*”) issues them a unique digital certificate that would be required to receive or install the Right-To-Execute. Ex. 1016 at 43, 47-48, 49, 50.

C205. Registering a public key with a certificate authority and obtaining a digital certificate was a conventional well-known technique. See ¶¶ C81-C89, A148-A153, *above* (e.g., Ex. 1017 (Denning) at 170; Ex. 1048 (RFC 1422) at 8, 11-12).

C206. ABYSS shows that protected processors are part of computer systems and can communicate over a network. E.g., Ex. 1016 at 50. This means the protected processors necessarily have access to a network interface for communicating with remote devices.

C207. Therefore, ABYSS shows that each protected processor is associated with a network interface allowing it to communicate with a software vendor (“*means for . . . communicating with a master repository*”), and request and obtain a digital certificate (“*obtain[] an identification certificate*”).

C208. It is my opinion that ABYSS in view of Denning renders claims 24, 51, and 81 of the ’859 patent obvious.

2. ABYSS Would Have Made the ’072 Patent Obvious

C209. The independent claims the ’072 patent (claims 1, 10, and 18) describe similar processes. Each independent claim specifies a “document platform” that requests and/or receives content and “at least one usage right” from

a “document repository.” The document platform stores the content and the usage right in separate files, and it renders the content but only in accordance with the usage right (or rights). Claims 10 and 18 each additionally specify that the document repository authenticates the document platform. Claim 10 additionally specifies that the document repository stores the content and the at least one usage right in separate files.

C210. As I explained above (¶ B116-B123), a digital document is a “composite digital work,” which is a digital work composed of other digital works, each with its own usage rights. While ABYSS and Denning do not explicitly describe a “digital document,” Hartrick does. It would have been obvious to use the ABYSS infrastructure to distribute Hartrick’s softcopy books. *See* ¶¶ C140-C145, *above*; ¶¶ E74-E98, *below*.

C211. ABYSS discloses the processes described in the ’072 independent claims. ABYSS explains how protected processors can be used to provide a software lending library (or a book lending library when combined with Hartrick), where a protected processor on a client workstation (“document platform”) can request a copy of an application and a Right-To-Execute (“retrieving . . . a digital document and at least one usage right”) from a file server’s protected processor (“document repository”). The file server’s protected processor receives the request (“receiving a request”), authenticates the client workstation, checks whether copies

of the application are available, and if so, will provide the workstation with a copy of the application and the Right-To-Execute (“transferring the digital document”).

The Right-To-Execute contains terms and conditions of use, and the Right-To-Execute and application are separate files (“storing . . . in separate files”). The workstation’s protected processor will verify it has authorization to install or receive the content before installing/storing it (“determining by the document platform”). The workstation’s protected processor will execute the software in accordance with the software’s terms and conditions of use.

a. Claim 1 of the '072 Patent

C212. The processes I described above disclose all the limitation of claim 1 of the '072 patent. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'072 Patent: Claim 1</u>	<u>Relevant Concepts and Paragraphs</u>
1. A method for securely rendering digital documents, comprising:	It would have been obvious to distribute <u>Hartrick</u> ’s softcopy books using the ABYSS system.
[1.a] retrieving, by a document platform, a digital document and at least one usage right associated with the digital document from a document repository, the at least one usage right specifying a manner of use indicating the manner in which the digital document can be rendered;	<p><u>“retrieving . . . a digital document”</u></p> <p>A workstation’s protected processor requests and receives an application from a server protected processor. ¶¶ C46-C54</p> <p><u>“digital document”</u></p> <p>ABYSS could be used to distribute</p>

	<p>softcopy books as described in <u>Hartrick</u>. ¶¶ C140-C145</p> <p><u>“document platform”</u></p> <p>Protected processor on a workstation. ¶¶ C23-C59</p> <p><u>“at least one usage right”</u></p> <p>The Right-To-Execute would contain the specialized SGML tags for each chapter of the book. ¶¶ C146-C152</p> <p><u>“document repository”</u></p> <p>File server’s protected processor. ¶¶ C16; C32-C45; C153-C170</p>
[1.b] storing the digital document and the at least one usage right in separate files in the document platform;	Softcopy book and Right-To-Execute (containing the SGML tags) are separate files. ¶¶ C35; C140-C145
[1.c] determining, by the document platform, whether the digital document may be rendered based on the at least one usage right; and	Workstation’s protected processor checks terms and conditions of use before permitting execution or rendering. ¶¶ C5-C13, C42-C44; C73, C146-C152
[1.d] if the at least one usage right allows the digital document to be rendered on the document platform, rendering the digital document by the document platform.	Workstation’s protected processor checks terms and conditions of use before permitting execution or rendering. ¶¶ C5-C13, C42-C44; C146-C152

b. Claim 10

C213. The processes I described above disclose all the limitation of claim 10 of the ’072 patent. I note that Claim 10.b specifies a document repository receives

a “request for access” to content and in response it “transfers” content to the document platform. Claims 1 and 18 each specify a “request to transfer.” A person of ordinary skill would understand that a “request for access” to content would be satisfied by a “request to transfer” content. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'072 Patent: Claim 10</u>	<u>Relevant Concepts and Paragraphs</u>
10. A method for securely rendering digital documents, comprising:	<i>See</i> Claim 1.
[10.a] storing a digital document and at least one usage right in separate files in a document repository, wherein the at least one usage right is associated with the digital document;	Softcopy book and Right-To-Execute (containing the SGML tags) are separate files. ¶¶ C35; C140-C145
[10.b] receiving a request from a document platform for access to the digital document;	A file server’s protected processor receives requests from workstations to borrow a copy of an application or book. ¶¶ C23-C59
[10.c] determining, by the document platform, whether the request may be granted based on the at least one usage right, the determining step including authenticating the document platform and determining whether the at least one usage right includes a manner of use that allows transfer of the digital document to the document platform;	<p><i>See</i> Claim 1.c.</p> <p><u>“authenticating the document platform”</u></p> <p>In public-key configuration, file server will authenticate the workstation’s protected processor’s digital certificate. ¶¶ C29, C51; C86-C122</p> <p><u>“a manner of use that allows transfer”</u></p> <p>File server will determine whether copies of the application or book and Right-To-Execute are available. ¶¶ C5-C13, C37, C73, C106-C126</p>

	Workstation determines whether the Right-To-Execute can be installed. ¶¶ C104-C105, C124
[10.d] if the at least one usage right allows the transfer of the digital document to the document platform, transferring the digital document and the at least one usage right associated with the digital document to the document platform;	File server will determine whether copies of the application or book and Right-To-Execute are available. ¶¶ C5-C13, C37, C73, C106-C126
[10.e] storing the digital document and the at least one usage right in the document platform, wherein the at least one usage right is stored in a separate file from the digital document; and	See Claim 1.b.
[10.f] rendering the digital document by the document platform.	See Claim 1.d.

c. Claim 18

C214. The processes I described above disclose all the limitation of claim 18 of the '072 patent. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'072 Patent: Claim 18</u>	<u>Relevant Concepts and Paragraphs</u>
18. A method for securely rendering digital documents, comprising:	See Claim 1.
[18.a.] receiving a request from a document platform to transfer a digital document from a document repository to the document platform, the digital document having at least one usage right associated therewith;	See Claim 10.b.
[18.b.] authenticating the document	See Claim 1.c & 10.c.

<p>platform by accessing at least one identifier associated with the document platform or with a user of the document platform and determining whether the identifier is associated with at least one of the document platform and a user authorized to use the digital document;</p>	<p><u>“authenticating . . . by accessing at least one identifier”</u></p> <p>In public-key configuration, the identifier in the digital certificate associated with the workstation’s protected processor. ¶¶ C29, C51; C86-C122</p> <p><u>“determining whether the identifier . . . is associated with . . . [an] authorized [document platform or user]”</u></p> <p>In public-key configuration, server’s protected processor will validate the workstation’s digital certificate by determining if it was issued by the group’s certificate authority. ¶¶ C29, C51; C86-C122</p>
<p>[18.c.] if the authenticating step is successful, determining whether the digital document may be transferred and stored on the document platform based on a manner of use included in the at least one usage right;</p>	<p>File server will determine whether copies of the application and Right-To-Execute are available. ¶¶ C5-C13, C37, C73, C106-C126</p>
<p>[18.d.] if the at least one usage right allows the digital document to be transferred to the document platform, transferring the digital document and the at least one usage right associated with the digital document to the document platform;</p>	<p><i>See Claim 10.d.</i></p>
<p>[18.e.] storing the digital document and the at least one usage right in the</p>	<p><i>See Claim 10.e.</i></p>

document platform, wherein the at least one usage right is stored in a separate file from the digital document;	
[18.f.] determining, by the document platform, whether the digital document may be rendered based on the at least one usage right; and	<i>See Claim 1.c.</i>
[18.g.] if the at least one usage right allows the digital document to be rendered on the document platform, rendering the digital document by the document platform	<i>See Claim 1.d.</i>

C215. It is my opinion that ABYSS in view of Denning renders claims 1, 10, and 18 of the '072 patent obvious.

d. Claims 8, 16, and 24

C216. Claims 8, 16, and 24 depend from claims 1, 10, and 18, respectively, and specify that “at least one part of the digital document and the at least one usage right are stored on a same device.”

C217. ABYSS shows the protected process is installed into a computer system, and that the application and Right-To-Execute are both stored on that computer system (*i.e.*, “*a same device*”).

C218. It is my opinion that ABYSS in view of Denning renders claims 8, 16, and 24 of the '072 patent obvious.

3. The Claims of the '956 and '007 Patents

C219. I am addressing the claims of the '956 and '007 patents together.

Both the '956 and '007 claims are directed to the same basic processes for transferring content between a sending device and recipient device. The '956 claims are directed toward client or recipient device, while the '007 claims are directed toward the server or sending device.

a. Claims 1, 7, and 13 ('956) and 1, 6, and 11 ('007)

C220. ABYSS discloses the processes described in the '956 and '007 independent claims. Because the independent claims are highly similar, I have addressed them together below. In the tables below, I have mapped each claim element to the relevant paragraphs of my analysis.

C221. ABYSS explains how protected processors can be used to provide a software lending library, where a protected processor on a client workstation (“recipient computing device” or “apparatus”) can request a copy of an application and a Right-To-Execute from a file server’s protected processor (“sending computing device”). In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[Claim 1. preamble] A computer-implemented method of rendering digital content by at least one	[Claim 1. preamble] A computer-implemented method of distributing digital content to at least	ABYSS describes methods, systems, and software for the trusted distribution of

recipient computing device in accordance with usage rights information, the method comprising:	one recipient computing device to be rendered by the at least one recipient computing device in accordance with usage rights information, the method comprising:	software. <u>Sending computing device</u> Lending library file server; software vendor's server ¶¶ C106-C126 <u>Recipient computing device</u> Workstation containing a protected processor ¶¶ C106-C126
[Claim 7. preamble] A recipient apparatus for rendering digital content in accordance with usage rights information, the recipient apparatus comprising:	[Claim 6. preamble] A sending apparatus for distributing digital content to at least one recipient computing device to be rendered by the at least one recipient computing device in accordance with usage rights information, the sending apparatus comprising:	<u>Sending computing device</u> Lending library file server; software vendor's server. ¶¶ C106-C126 <u>Recipient computing device</u> Workstation containing a protected processor ¶¶ C106-C126
[Claim 13. preamble] At least one non-transitory computer-readable medium storing computer-readable instructions that, when executed by at least one recipient computing device, cause the at least one recipient computing device to:	[Claim 11. preamble] At least one non-transitory computer-readable medium storing computer-readable instructions that, when executed by at least one sending computing device, cause the at least one sending computing device to:	<u>Sending computing device</u> Lending library file server; software vendor's server ¶¶ C106-C126 <u>Recipient computing device</u> Workstation

		containing a protected processor ¶¶ C106-C126
--	--	--

C222. ABYSS discloses systems, methods, and software for implementing the techniques described in the article. As I explained above, the protected processor contains a processor, memory, and software for enabling its functionality. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
<p>[7.a.] one or more processors; and</p> <p>[7.b.] one or more memories operatively coupled to at least one of the one or more processors and having instructions stored thereon that, when executed by at least one of the one or more processors, cause at least one of the one or more processors to:</p>	<p>[6.a.] one or more processors; and</p> <p>[6.b.] one or more memories operatively coupled to at least one of the one or more processors and having instructions stored thereon that, when executed by at least one of the one or more processors, cause at least one of the one or more processors to:</p>	<p><u>“rendering device”</u></p> <p>computer system with a protected processor. See ¶¶ C23-C59; C76-C126</p> <p>Each protected processor contains a processor and memory</p>

C223. ABYSS explains that, when the workstation’s protected processor sends the request to the file server’s protected processor, the file server validates a digital certificate, authenticates the client workstation, and checks whether the

client is authorized, for example, by validating a digital certificate or a signed message (“determining if the . . . recipient computing device is trusted to receive the digital content”). If the client is authorized, the file server checks whether copies of the application are available, and if so, will provide the workstation with a copy of the application and the Right-To-Execute. The client workstation’s protected processor receive the application and Right-To-Execute (“receiving the digital content”). In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>’956 Patent</u>	<u>’007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[1.a.] receiving the digital content by the at least one recipient computing device from at least one sending computing device only if the at least one recipient computing device has been determined to be trusted to receive the digital content from the at least one sending computing device;	[1.a.] determining, by at least one sending computing device, if the at least one recipient computing device is trusted to receive the digital content from the at least one sending computing device;	<p><u>“determination of trust”</u></p> <p>In public-key configuration, validating a protected processor’s digital certificate. ¶¶ C81-C89, C118-C126.</p> <p><u>trusted “repository”</u></p> <p>Workstation with a protected processor. ¶¶ C23-C45; C153-C170</p>
[7.c.] enable the receipt of the digital content by the recipient apparatus from at least one sending computing	[6.c.] determine if the at least one recipient computing device is trusted to receive the digital	<p><u>“if ... trusted”</u></p> <p>See element a, above</p>

device only if the recipient apparatus has been determined to be trusted to receive the digital content from the at least one sending computing device;	content from the sending apparatus;	<u>trusted “repository”</u> Workstation with a protected processor. ¶¶ C23-C59; C153-C170.
[13.a.] receive the digital content from at least one sending computing device only if the at least one recipient computing device has been determined to be trusted to receive the digital content from the at least one sending computing device;	[11.a.] determine if the at least one recipient computing device is trusted to receive the digital content from the at least one sending computing device;	A copy of the application and its Right-To-Execute is transferred to the workstation ¶¶ C81-C89, C118-C126

C224. As I just explained, if the file server determines the client is authorized and that copies of the application are available, the file server will provide the workstation’s protected processor with a copy of the application (“sending the digital content”) and the Right-To-Execute (“sending the usage information”). In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>’956 Patent</u>	<u>’007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
N/A	[1.b.] [11.b.] send[ing] the digital content, by the at least one sending computing device, to the at least one recipient computing device only if the at least one recipient computing device has been determined to be trusted to	A copy of the application and its Right-To-Execute is transferred to the workstation

	<p>receive the digital content from the at least one sending computing device; and</p> <p>[6.d.] send the digital content, by the sending apparatus, to the at least one recipient computing device only if the at least one recipient computing device has been determined to be trusted to receive the digital content from the sending apparatus; and</p>	¶¶ C81-C89, C118-C126
	<p>[1.c.] [11.c.] send[ing] usage rights information indicating how the digital content may be rendered by the at least one recipient computing device, the usage rights information being enforceable by the at least on recipient computing device.</p> <p>[6.e.] send usage rights information indicating how the digital content may be rendered by the at least one recipient computing device, the usage rights information being enforceable by the at least on recipient computing device.</p>	<p>A copy of the application and its Right-To-Execute is transferred to the workstation. ¶¶ C34-C38, C69-C71, C81-C89, C118-C126</p>

C225. When a user attempts to execute the borrowed software application (or any application), the protected processor will receive a request to access the Right-To-Execute (“receiving a request . . . to render the digital content”). The user’s protected processor will determine whether the user possesses a Right-To-Execute, and whether the terms and conditions are satisfied (“determining . . .

whether the digital content may be rendered”). Ex. 1016 (ABYSS) at 40, 44-45, 49. If they are, the protected processor will permit the application to execute as normal (“rendering the digital content”). *Id.* If they are not, for example because the loan period has expired, then the protected processor will not allow execution. *Id.* In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[1.b.], [7.d], [13.b] receiv[ing], [by the at least one recipient computing device,] a request to render the digital content;	N/A	User requests to execute an application. ¶¶ C55-C67, C146-C152
[1.c.], [7.e], [13.c] determin[ing], based on the usage rights information, whether the digital content may be rendered by the at least one recipient computing device [apparatus]; and		Protected processor enforcing terms and conditions of use. ¶¶ C5-C13, C42-C44, C55-C67
[1.d.], [7.f], [13.d] render[ing] the digital content, [by the at least one recipient computing device,] only if it is determined that the content may be rendered by the at least one recipient computing device [apparatus].		¶¶ C5-C10, C55-C67, C146-C152

C226. Therefore, it is my opinion that ABYSS in view of Denning renders claims 1, 7, and 13 of the '956 patent obvious. And that ABYSS in view of Denning renders claims 1, 6, and 11 of the '007 patent obvious.

b. Claims 2, 8, 14 ('956)

C227. Claims 2, 8, and 14 of the '956 depend from claims 1, 7, and 13, respectively, and each specifies that the recipient device will “*deny[] the request and prevent[] rendering of the digital content . . . if it is determined that the digital content may not be rendered . . .*”

C228. As I have previously stated, when a user attempts to execute a software application, the user’s protected processor will permit execution if the terms and conditions in the Right-To-Execute are satisfied, and it will deny the request if they are not. Without access to the Right-To-Execute, the user will be unable to execute the software. *See ¶¶ C34-C45, above.*

c. Claims 3, 9, 15 ('956) and 2, 7, 14 ('007)

C229. Claims 3, 8, and 15 of the '956 depend from claims 1, 7, and 12, respectively, and each specifies the usage rights information includes a “*condition under which the content can be rendered.*” Similarly, claims 2, 7, and 14 of the '007 depend from claims 1, 6, and 12, respectively, and each specifies the usage rights information includes a “*condition under which the content can be rendered.*”

C230. As I have previously stated, each Right-To-Execute can contain conditions such as an expiration date or a maximum number of uses. *See ¶¶ C41-C45, C149, above.*

d. Claims 4, 10, 16 ('956) and 3, 8, 15 ('007)

C231. Claims 4, 10, and 15 of the '956 depend from claims 1, 7, and 13, respectively, and each specifies that the step of “*receiving the digital content comprises*” two additional steps by the recipient device: requesting an “authorization object” and “receiving” it if authorized.

C232. Claims 3, 8, and 16 of the '007 depend from claims 1, 6, and 11, respectively, each and specifies that the step of making “*the determination of trust*” comprises two additional steps by the sending device: receiving a “request . . . for an authorization object” and transmitting the object “when it is determined that the request should be granted.”

C233. ABYSS describes an “authorization process” run by a protected processor that enables the protected processor obtain and receive authorization from a software vendor through a “cryptographic transaction.” As I explained above, this authorization could take the form of a group supervisor key, a digital certificate, or a digitally signed message from the software vendor or an authorized representative of the vendor. *See ¶¶ C48-C54, C104-C105, above.* I also explained why this object would be required for a workstation’s protected processor to receive and use applications from a file server’s protected processor. *See ¶¶ C48-C54, C104-C105, above.*

C234. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[4.a.], [10.a.], [16.a.] requesting an authorization object for the at least one recipient computing device [receiving apparatus] to make the digital content available for use, the authorization object being required to receive the digital content and to use the digital content; and	[3.a.], [8.a.], [16.a.] receiving a request from at least one recipient computing device for an authorization object required to render the digital content; and	Requesting a digitally signed receipt from the software vendor to join a lending library ¶¶ C23-C31, C48-C54, C104-C105, C114-C117
[4.b.], [10.b.], [16.b.] receiving the authorization object if it is determined that the request for the authorization object should be granted.	[3.b.], [8.b.], [16.b.] transmitting the authorization object to the at least one recipient computing device when it is determined that the request should be granted.	Receiving/sending a digitally signed receipt from the software vendor to join a lending library ¶¶ C23-C31, C104-C105, C114-C117

C235. Therefore, it is my opinion that ABYSS in view of Denning renders claims 4, 10, and 16 of the '956 patent obvious.

C236. It is my opinion that ABYSS in view of Denning renders claims 3, 8, and 16 of the '007 patent obvious.

e. **Claims 5, 11, 17 ('956) and Claims 4, 9, 14 ('007)**

C237. Claims 5, 11, and 17 of the '956 patent depend from claims 1, 7, and 13, respectively, each specifies a method, apparatus, or software where the step of “receiving the digital content” or “enabling the receipt of the digital content” comprises three additional steps by the recipient computing device: generating a “registration message,” exchanging a session key with the sending device, and conducting a session where it receives content from the sending device.

C238. Claims 4, 9, and 14 of the '007 patent depend from claims 1, 6, and 13, respectively, each specifies a method, apparatus, or software where the step of making “the determination of trust” comprises four additional steps by the sending computing device: receiving a “registration message” from a recipient device, validating the authenticity of the recipient device, exchanging a session key with the recipient computing device, and conducting a session where it sends content to the recipient device.

C239. As with the independent claims, these '956 and '007 claims are directed to the same process, but the '956 claims cover the client device and the '007 claims cover the server device. These additional steps simply add standard cryptographic techniques that were well-known in the art.

C240. ABYSS explains that protected processors can “set up a cryptographically secure channel between them, mediated by a session key” and

that “[t]here are a number of standard ways to do this.” Ex. 1016 (ABYSS) at 42.

As I explained above, it would have been obvious to configure the protected processors with a key management strategy based on public-key cryptography.

C241. In the software lending library example, a workstation’s protected processor would initiate a connection with the file server’s protected processor through a standard certificate exchange process. When configured according to Denning, this message (“registration message”) will include the workstation’s protected processor’s digital certificate signed by a trusted authority (“identification certificate”), a session key (“exchanging messages including . . . [a] session key”), and a timestamp (“registration identifier”). See ¶¶ C81-C89, A148-A153, *above*; Ex. 1017 (Denning) at 178-179, 175.

C242. The file server’s protected processor will validate the workstation’s digital certificate and take other steps to determine whether the workstation is authorized, and establish a secure connection using the session key (“conducting a secure transaction using the session key”). If the file server determines the workstation is authorized and that copies of the application are available, it will transmit a copy of the application and a Right-To-Execute to the workstation’s protected processor through the secure channel (“sending [receiving] the digital content”).

C243. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[5.a.], [11.a.], [17.a.] generating a registration message, the registration message including an identification certificate of the recipient computing device [apparatus] and a random registration identifier, the identification certificate being certified by a master device;	[4.a.] [9.a.] [14.a.] receiving a registration message from the at least one recipient device, the registration message including an identification certificate of the recipient computing device and a random registration identifier, the identification certificate being certified by a master device;	In public-key configuration, standard certificate exchange process ¶¶ C81-C89, A148-A153
N/A	[4.b.] [9.b.] [14.b.] validating the authenticity of the at least one recipient device;	Standard certificate exchange process ¶¶ C81-C89, A148-A153
[5.b.], [11.b.], [17.b.] exchanging messages including at least one session key with at least one provider computing device, the session key to be used in communications during a session; and	[4.c.] [9.c.] [14.c.] exchanging messages including at least one session key with the at least one recipient device, the session key to be used in communications; and	Standard certificate exchange process ¶¶ C81-C89, A148-A153
[5.c.], [11.c.], [17.c.] conducting a secure transaction using the session key, wherein the secure	[4.d.] [9.d.] [14.d.] conducting a secure transaction using the session key, wherein the secure	Standard certificate exchange process and secure sessions

transaction includes receiving the digital content.	transaction includes sending the digital content to the at least one recipient device.	¶¶ C31, C81-C89, A148-A153
---	--	----------------------------

C244. It is my opinion that ABYSS in view of Denning renders claims 5, 11, and 17 of the '956 patent obvious.

C245. It is my opinion that ABYSS in view of Denning renders claims 4, 9, and 14 of the '007 patent obvious.

f. Claims 6, 12, 18 ('956) and Claims 5, 10, 15 ('007)

C246. Claims 6, 12, and 18 of the '956 patent depend from claims 5, 11, and 17, respectively, each specifies the method, apparatus, or software comprises two additional steps. The two additional steps in each of claims 6, 12, and 18 are identical. These steps simply add standard cryptographic techniques that were well-known in the art.

C247. Claims 5, 10, and 15 of the '007 patent depend from claims 1, 9, and 14, respectively, each specifies a method, apparatus, or software where the step of making “validating” comprises four additional steps. The four additional steps in each of claims 5, 10, and 15 are identical.

C248. As with the independent claims, these '956 and '007 claims are directed to the same process, but the '956 claims cover the client device and the '007 claims cover the server device. These additional steps simply add standard cryptographic techniques that were well-known in the art.

C249. As I explain above, Denning shows the process of establishing a secure channel can include a handshake process. *See ¶¶ C81-C89, A148-A153, above* (Ex. 1017 (Denning) at 175, 178-79). In the software lending library, the file server’s protected processor will receive the message from the workstation that contains a digital certificate, session key, and timestamp. The file server will validate the digital certificate (“verifying the identification certificate”), and then take additional steps to ensure the connection is secure and that the workstation is authentic and on-line. The file server can do so by generating a random number (“generating [receiving] a message . . . including a nonce”), encrypting it with the session key or the workstation’s public key, and transmitting it to the workstation’s protected processor (“sending [receiving] a message to test authenticity”). The workstation’s protected processor will decrypt the random number, alter it, and sent it back to the file server (“processing the generated message”). The file server verifies the message was processed correctly, and if it was, that indicates that the workstation is authentic (“verifying” the message was “correctly processed”). *See ¶¶ C81-C89, A148-A153, above.*

C250. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>’956 Patent</u>	<u>’007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
N/A	[5.a.] [10.a.] [15.a.]	In public-key

	verifying the identification certificate of the at least one recipient device	configuration, this will be validated during the standard certificate exchange process. ¶¶ C81-C89, A148-A153
[6.a.], [12.a.], [18.a.] receiving a message to test the authenticity of the at least one recipient computing device [apparatus], the generated message including a nonce; and	[5.b.] [10.b.] [15.b.] generating a message to test the authenticity of the at least one recipient device, the generated message including a nonce; [5.c.] [10.c.] [15.c.] sending the generated message to the at least one recipient device; and	A standard handshake ¶¶ C81-C89, A148-A153 (Ex. 1017 at 175)
[6.b.], [12.b.], [18.b.] processing the generated message to indicate authenticity.	[5.d.] [10.d.] [15.d.] verifying if the at least one recipient device correctly processed the generated message.	A standard handshake ¶¶ C81-C89, A148-A153 (Ex. 1017 at 175)

C251. It is my opinion that ABYSS in view of Denning renders claims 6, 12, and 18 of the '956 patent obvious.

C252. It is my opinion that ABYSS in view of Denning renders claims 5, 10, and 15 of the '007 patent obvious.

4. Analysis of the Claims of the '576 Patent

a. Claim 18 of the '576 Patent

C253. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'576 Patent: Claim 18</u>	<u>Relevant Concepts and Paragraphs</u>
<p>18. A method for controlling rendering of digital content on an apparatus having a rendering engine configured to render digital content and a storage for storing the digital content, said method comprising:</p>	<p><u>“digital content”</u></p> <p>Software. ¶¶ C6-C10</p> <p><u>“apparatus”</u></p> <p>computer system. ¶¶ C6-C17</p> <p><u>“rendering engine”</u></p> <p>protected processor. ¶¶ C6-C31</p> <p><u>“renders”</u></p> <p>Executing software. ¶¶ C55-C67</p> <p><u>“storage for storing digital content”</u></p> <p>storage device connected to the computer system. ¶¶ C6-C17; C153-C170</p> <p><u>“controlling rendering of digital content on an apparatus”</u></p> <p>protected processor enforcing a Right-to-Execute containing a decryption key. ¶¶ C32-C45; C146-C152</p>
<p>[18.a.] specifying rights within said apparatus for digital content stored in said storage, said rights specifying how digital content can be rendered;</p>	<p><u>“digital content”</u></p> <p>Software. ¶¶ C6-C10</p> <p><u>“rights”</u></p> <p>Terms and conditions of use in Right-</p>

	<p>To-Execute. ¶¶ C19; C32-C45; C57; C65; C69; C146-C152</p> <p><u>“said rights specifying how digital content can be rendered”</u></p> <p>Right-to-Execute contains data specifying if the application can be executed, transferred, or lent out. ¶ C150; <i>see generally</i> ¶¶ C32-C45; C146-C152</p> <p><u>“within said apparatus”</u></p> <p>Right-to-Execute is stored in the protected processor. ¶¶ C146-C170</p>
[18.b.] storing digital content in said storage	
[18.c.] receiving a request for rendering of said digital content stored in the storage;	<p><u>“receiving a request for rendering”</u></p> <p>User attempts to execute an application, sending a request to the protected processor. ¶¶ C46-C47; C73; C107</p>
[18.d.] checking whether said request is for a permitted rendering of said digital content in accordance with said rights specified within said apparatus;	<p>Protected processor checks whether the Right-To-Execute may be used and also checks conditions such as an expiration time and date and maximum allowed uses. ¶ C74; <i>see generally</i> ¶¶ C32-C45; C146-C152.</p>
[18.e.] processing the request to make said digital content available to the rendering engine for rendering when said request is for a permitted rendering of said digital content;	<p><u>“when said request is for a permitted rendering”</u></p> <p>Protected processor determines whether execution is permitted. ¶¶ C32-C45; C146-C152.</p> <p><u>“processing the request to make said</u></p>

	<p><u>digital content available to the rendering engine for rendering</u></p> <p>Protected processor uses the decryption key in the Right-To-Execute to decrypt and the application executes it. ¶¶ C6-C10 (decryption) C46-C47; C73; C107 (request)</p>
[18.f.] authorizing a repository for making the digital content available for rendering, wherein the digital content can be made available for rendering only by an authorized repository, the repository performing the steps of;	<p>An authorized workstation can join the software lending library by first registering with the software vendor and obtaining a digitally signed receipt. ¶¶ C79-C94 (certificate authority); C106-C126 (lending library)</p>
[18.g.] making a request for an authorization object [sic] required to be included within the repository for rendering of the digital content; and	<p><u>“required to be included within the repository for rendering of the digital content”</u></p> <p>Protected processor will register with the software vendor and obtain a digital certificate from the group certificate authority and obtain a digitally signed receipt. ¶¶ C79-C94 (certificate authority); C104-C105 (receipt)</p> <p><u>“making a request for an authorization object”</u></p> <p>Requesting a digitally signed receipt. C104-C105 (receipt).</p>
[18.g.] receiving the authorization object when it is determined that the request should be granted.	<p><u>“determines that the request should be granted”</u></p> <p>The vendor determines authorization should be granted and provides a digitally signed receipt. C104-C105</p>

	(receipt).
--	------------

C254. It is my opinion that ABYSSS in view of Denning renders claim 18 of the '576 patent obvious.

b. Claim 19 of the '576 Patent

C255. Claim 19 depends from claim 18 and specifies “the checking step comprises comparing a requested use with a use specified by the rights.”

C256. ABYSS shows each Right-To-Execute can specify whether execution, transfer, or lending is permitted. When a protected processor receives a request to execute, transfer, or lend software, it determines if the requested use is permitted in the Right-To-Execute (“*comparing a requested use with a use specified by the rights*”). Ex. 1016 at 38-40, 42-45, 50.

c. Claim 21 of the '576 Patent

C257. Claim 21 depends from claim 18 specifies “requesting a transfer of the digital content from an external memory to the storage.”

C258. ABYSS shows that software vendors and lending libraries store software (“*external memory*”), and a protected processor on a workstation can retrieve (“*requesting a transfer*”) a copy of the software and a Right-To-Execute and store them in memory (“*storage*”). Ex. 1015 at 47, 49, 50 (using “secure two-way communication between the software vendor and the user . . . [b]oth software,

and Rights-To-Execute can then be distributed . . . by download from host systems to workstations”).

C259. In addition, ABYSS shows that applications and Rights-To-Execute can be backed up by transferring backup copies to external memory such as a smartcard. Ex. 1016 at 43 (“Backup”), 45-46. To restore the files from the backup disk (“*external memory*”), the user can request to transfer the backup copies (“*requesting a transfer*”) back into the protected processor (“*storage*”). Ex. 1016 at 43 (“installation [of the backup] is accomplished by inserting the smart card into the replacement system’s slot . . . and typing: backup –install”), 45-46.

d. Claim 24 of the ‘576 Patent

C260. Claim 24 depends from claim 18 and specifies “*wherein the digital content is video content.*”

C261. A person of ordinary skill would have understood that the ABYSS could be used to protect content such as video. A computer that can play video content is a “video system.” ABYSS shows that a protected processor can be included in a “range of computing systems.” See ¶ C3. A person of ordinary skill would have understood that the protected processor could be included in computer systems that could play videos, and that any type of content could be rendered by the protected processor. See ¶¶ C134-C141, *above*.

e. Claim 25 of the ‘576 Patent

C262. Claim 25 depends from claim 18 and specifies “wherein the apparatus is a portable device” and “wherein the method is implemented in a portable device,” respectively.

C263. ABYSS specifically states that the solutions outlined therein are intended to “solve the problem of illicit duplication” across the spectrum of computing systems. Ex. 1016 at 38. A person of ordinary skill would have understood that this would include laptop computers, which are “portable computers.”

f. Claims 26 of the ‘576 Patent

C264. Claim 26 depends from claim 18 and specifies “wherein the rights are provided by a provider of the digital content.”

C265. ABYSS explains each Right-to-Execute contains terms and conditions (“*rights*”) that are specified by the software vendor (“*the provider of the digital content*”). Ex. 1015 at 47, 49, 50.

g. Claim 27 of the ‘576 Patent

C266. Claim 27 depend from claim 18 and species “wherein the means for processing [step] comprises transmitting the digital content to the rendering machine.”

C267. When a user requests to execute software, the protected processor determines the requested use (*e.g.*, executing) is permitted, and if so, the

application is transmitted to the protected processor for decryption and execution.

Ex. 1016 at 38-40, 42-45, 48-50.

h. Claim 29 of the ‘576 Patent

C268. Claim 29 depend from claim 18 and species “wherein the rendering engine is configured to render digital content into a medium that is not further protected by rights and the request is for rendering the digital content into a medium that is not further protected by rights.”

C269. A person of ordinary skill would have understood that any type of content, including books and images, could be rendered by the protected processor. *See ¶¶ C134-C145, above.* It would have been obvious to configure the protected processor permit users to print books or images.

i. Claim 32 of the ‘576 Patent

C270. Claim 32 depends from claim 18 and specifies “wherein the rights are embodied in software instructions which implement the use privileges for the rights.”

C271. The Right-To-Execute (“*rights*”) is embodied in software instructions used by the protected processor and allows a software vendor to grant users particular rights, such as the right to execute, transfer, or loan software (“*privilege*”).

j. Claim 34 of the ‘576 Patent

C272. Claim 34 recites the method of claim 18, “further comprises: requesting receipt of digital content stored externally; and receiving the digital content if it is permitted to receive the digital content.”

C273. As I explained above with respect to claim 21, a protected processor can request a copy of an application and a Right-To-Execute stored at a software vendor or a lending library. The protected processor will receive the application only if authorized, *e.g.*, has paid or is a member of the library (“*if it is permitted to receive the digital content*”). Ex. 1016 at 47, 49, 50.

VII. SECTION D: Kahn (Exhibit 1018) and Linn (Exhibit 1057)

A. Overview of Kahn (Ex. 1018)

D1. Kahn describes a system for distributing digital content. Kahn shows that digital content can be any form of content such as “conventional digital representations of works (books, papers, images, sounds, software),” as well as “any digital material which is capable of producing desired manifestations for a computer user.” Ex. 1018 (Kahn) at 1:11-13, 1:17-21; *id.* at 4:46, 22:59-60 (digital content can be video). Kahn explains that digital content is distributed as a “digital object.” *Id.* at 1:11-13.

1. The Repository System

D2. Kahn explains that content is distributed using a distributed system of “repositories.” “A ‘repository’ 1002 is a digital storage system into which digital objects 1004 may be placed and retained for possible subsequent retrieval.” Ex. 1018 (Kahn) at 5:61-63. Kahn explains the repository includes various server systems that work with the repository to provide users with access to content and to enforce terms and conditions associated with the content:

The repository may contain other related information 1006 as well as management systems 1008. Where appropriate, such information may be provided to an “information and reference server” 1010. The result of retrieving a digital object from a repository may be a performance of the digital object (e.g., execution of a program) or the digital object itself (e.g., the program). This is important in the case of digital

objects which are only intended to be performed by users (e.g., a video game) rather than making the object itself available. A performance of a digital object may be stored as a new digital object and there may be separate terms and conditions associated with it.

Ex. 1018 (Kahn) at 5:63-6:7.

D3. The Kahn repository system enables “holders of rights in digital objects to control terms and conditions under which they are accessed or performed by users in a network.” Ex. 1018 (Kahn) at 3:4-7.

D4. Kahn explains that the repositories store content “at locations accessible in the network using a storage technique which renders the digital objects secure against unauthorized access.” Ex. 1018 (Kahn) at 2:17-21; *id.* at 3:30-32. Each repository ensures that the digital objects are stored “in a secure manner that both restricts access and allows for later verification that the deposit[ed] [content] has not been altered.” *Id.* at 4:46-49; *see id.* at 8:27-29 (“The repository 36 holds copies of digital objects in a secure and verifiable manner and controls access to the objects.”). Because the repositories are servers, Kahn relies on server-side security techniques (*e.g.*, the servers are physically remote from users) to protect the content.

2. Digital Objects and Terms & Conditions of Use

D5. For each digital object stored in a repository, the repository stores “reference information” about that object, which includes: “registration of rights in

the digital object including performance of the object; **accesses to and uses of digital object; the terms and conditions for use of digital objects**; the ownership and transfer of rights to disseminate digital objects; links between different digital objects.” *Id.* at 2:40-47 (emphasis added); *id.* at 4:7-13, 11:16-28.

D6. The terms and conditions of use in the Kahn scheme define how the object may be rendered or used. *Id.* at 3:4-14, 4:7-13, 6:40-46. Kahn shows that the terms and conditions of use allow an author to specify one more actions that may be taken with respect to the content, such as “performing” (*i.e.*, rendering) the content, making copies of the content, or modifying the content. *Id.* at 5:66-6:5 (“The result of retrieving a digital object from a repository may be a performance of the digital object (e.g., execution of a program) or the digital object itself (e.g., the program).”), 6:40-46.

D7. For example, Kahn shows an author can specify various rights, such as the right to “use,” “perform,” “modify,” or “transmit” content:

The information and reference server, the repositories, and the rights management system are also potentially accessible by network users 14, who may wish to obtain, use, modify, transmit, perform, and enhance selected digital objects, or the results of operations performed by or on digital objects.

Ex. 1018 (Kahn) at 7:17-22, 25:29-31.

D8. Kahn shows the terms and conditions can specify conditions on how many times a particular right can be executed: “The user selects the terms and conditions desired [], including the number of terms (e.g., A user may buy the right to make **5 copies** of the object or to perform it **ten times**).”

D9. A repository will allow a user to access and render a digital object, only if the user agrees to be bound the terms and conditions of use set by the object’s owner. *Id.* at 3:9-13, 2: 56-59. For example, Kahn explains:

Another general aspect of the invention concerns enabling holders of rights in digital objects to control terms and conditions under which they are accessed or performed by users in a network. Information is stored about terms and conditions for access to and performance of each digital object. The information is made available to a user in connection with a request for access to a digital object. The user is enabled to indicate assent to the terms and conditions. Access is permitted to the user only upon the user indicating assent to the terms and conditions.

Id. at 3:4-13; *id.* at 6:25-45 (“The terms and conditions in the properties record may allow the user to select which type of action to allow (e.g., retrieve object or perform object).”).

D10. Kahn shows that content is stored in a separate file from the terms and conditions of use. Kahn shows that a digital object contains an external properties

record, which contains the “stated terms and conditions for access and usage of the object.” Ex. 1018 (Kahn) at 6:14-16, 6:29-46.

D11. Kahn allows for nested digital objects. As Kahn explains:

Digital objects are "typed". Thus one can tell in a concatenated sequence of digital objects what kind of digital object is present (e.g., the "object itself" or a "performance of the object") and where each digital object starts and ends. One simple way to accomplish this is to include a type field as the first part of the sequence of binary digits which contains the necessary information. It could also be externally maintained (e.g., in a properties record, described below).

Id. at 6:8-16.

D12. As this passage explains, where several digital works are contained within one object, a header record is inserted at the beginning of the object that indicates where each work starts and ends.

D13. For nested objects, Kahn shows that instead of attaching the reference data to a header record, an external property record could be used. As Kahn explains:

The properties record may contain entries such as the identity of a rights management system 1018 (i.e., the system that has control over transfers of and compensation for rights in that object), the handle 1012 for that object, the originator of the object 1020, the name of the object (if any) 1022, a description of any work or other information or material incorporated in the object 1024, the time and date of deposit

1026, format information 1028, and stated terms and conditions for access and usage of the object 1030. The terms and conditions in the properties record may allow the user to select which type of action to allow (e.g., retrieve object or perform object).

Id. at 6:32-43.

3. Features of the Distributed System

D14. The repository interacts with a rights management system (for creating custom terms and conditions), a handle server (enables a user to search for content), and a payment server (to clear purchases). *Id.* at 7:28-36, 8:20-30, 8:66-9:19, 22:28-34.

D15. When an author uploads content to a repository, the author can specify standard terms and conditions that a user can purchase. These terms and conditions are enforced by the repository. *Id.* at 8:25-27 (“The RMS may delegate to the repositories the responsibility to handle simple terms and conditions.”); *see id.* at 22:26-26:39 (describing process for retrieving content from a repository under the standard terms and conditions).

D16. If a user wants to purchase different terms and conditions, the repository will refer the user to a “rights management system” that allows a user to negotiate with the author for custom terms and conditions. *Id.* at 6:55-57 (“The ‘rights management system’ 1038 is a system used to negotiate access rights and other rights not otherwise specified in the properties record.”), 8:20-25, 27:9-12

(“The RMS enters into a mixed initiative dialog 840 with the user to determine what terms and conditions are mutually acceptable, if any. This may also entail human interaction.”); *see generally id.* at 26:39-28:21 (explaining the process that allows a user to negotiate with the author for additional rights).

D17. The author may or may not grant the user’s request. *Id.* at 27:9-11 (“The RMS enters into a mixed initiative dialog 840 with the user to determine what terms and conditions are mutually acceptable, **if any**” (emphasis added)).

D18. A registered user can search for a digital work, and can purchase the work and a set of usage rights to allow the user to render the work. Ex. 1018 (Kahn) at 4:25-44; *id.* at 3:53-4:16; *id.* at 7:17-22; *id.* at 22:26 (a user can “obtain[] a digital object from a repository.”).

D19. Each digital object is identified by a “handle,” which is “an associated unique identifier” and can be used to locate an object stored in a repository. Ex. 1018 (Kahn) at 1:12-13; *id.* at 2:57-62 (“Another general aspect of the invention concerns enabling users of a network to access or perform digital objects stored in the network. There are multiple pointer servers each of which accepts identifiers of a subset of the digital objects and returns corresponding pointers to the locations of the digital objects in the network.”).

D20. “There may be multiple repositories,” and a user can search and access content from any of the repositories. *Id.* at 3:54-67. Kahn explains:

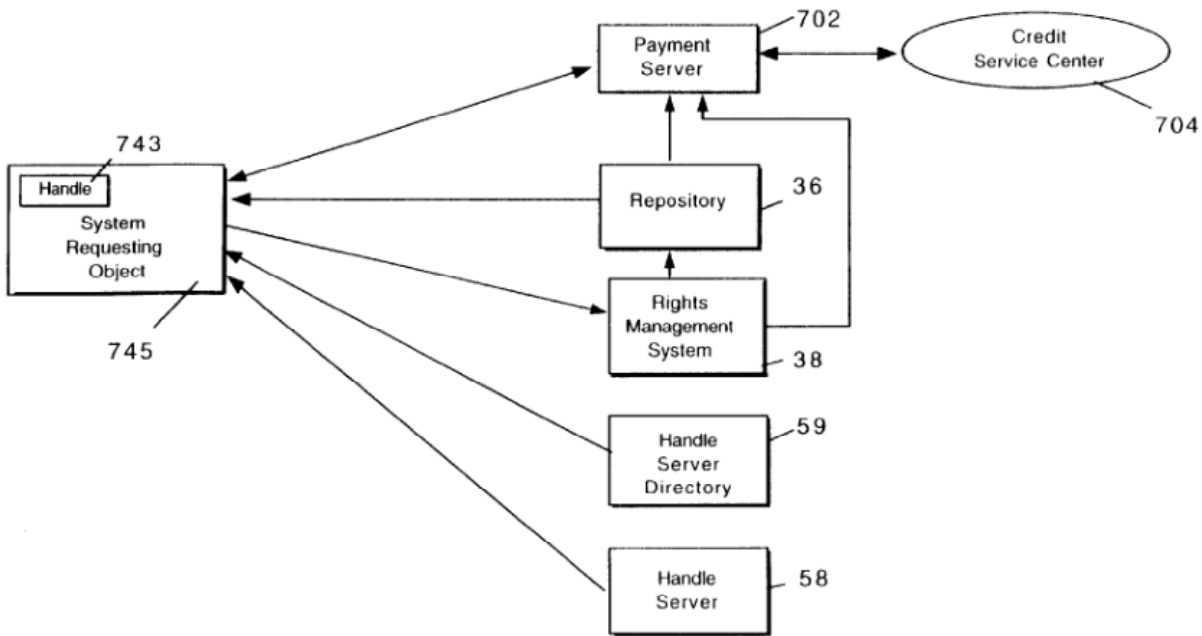
The system requesting the digital object obtains . . . a handle 743 for an object (742). . . The requesting system sends a request-for-pointer-information UDP packet 748 to the handle server. One or more pointers, once returned, identifies the network location of the one or more repositories (if one is associated with the object) and one or more rights management system, if one is associated with the object. This strategy assures a random distribution of handle server requests among many handle servers distributed on a network without a central nodal point in the system (for reliability).

* * *

An object may be stored in several repositories. Multiple pointers to these repositories may be returned to the requesting system. For each pointer returned by the handle server, the requesting system uses the pointer to attempt to obtain a copy of the digital object 762. If a copy is successfully obtained from one repository, the rest of the pointers will generally be ignored. If the requesting system cannot obtain the object from any of the repositories, it informs the user that it is unable to retrieve the object 764.

Id. at 23:65-24:33.

D21. An example of the interactions among these devices during a purchase transaction is depicted in Figure 23 of Kahn:



Ex. 1018 (Kahn) at Fig 23; *see also id.* at Figs. 24-27 (purchase transaction).

4. Cryptographic Security

D22. The repositories use public-key technologies such as digital certificates, digital signatures, and privacy enhanced mail to provide system integrity, privacy, authentication, and secure communications. Ex. 1018 (Kahn) at 5:6-8 (“The digital signature, privacy enhanced messaging, and other protection mechanisms assure the integrity of the system.”); *id.* at 9:37-43 (“As objects and other information are transmitted between the various system components, the privacy of the information must be ensured. The system uses available public key and digital signature technology to handle privacy and authentication in the system as follows.”).

D23. Kahn explains how the RSA algorithms are used to provide privacy and authentication in the system:

Three researchers at MIT, Rivest, Shamir and Adelman, later developed a pair of functions meeting the requirements specified by Diffie and Hellman. These functions are now known as the RSA algorithms. There are also other known ways to implement public key algorithms.

Since either key of a public key cryptography pair can be used to perform the initial encryption, an interesting effect can be achieved by using the secret key of the pair to encrypt messages to be sent.

Anyone with access to the public key can decrypt the message and on doing so successfully, knows that the message must have been sent by the person holding the corresponding secret key. This use of the secret key acts like a signature, since the decryption only works with the matching public key. If the public key for the sender is stored in a public directory, any recipient can verify the identity of the sender.

Id. at 9:61-10:9.

D24. Kahn explains that digital signatures are used to validate the integrity of digital objects:

As shown in FIG. 3, **the private key 100 can also be used to prove that an object 102 has not been altered by anyone after the object's rights holder (e.g., an author) has fixed the representation of the object.** If the author performs a hash 104 over all of the object's bits, and then encrypts 106 the hash value with his secret key 100 to produce a digital signature 105, **a recipient can decrypt the hash**

value, rehash the original object and compare the two hash values to ensure that the object has not been altered.

Id. at 10:10-17 (emphasis added).

D25. Kahn explains that digital certificates and digital signatures are used for authentication and to identify authorized users. Kahn explains that certificate authorities are used to authenticate entities participating in the system. As Kahn explains:

One problem that must still be addressed is knowing whether a public key 108 found in a directory for a given correspondent is valid or a bogus key inserted by a malicious person. **One way to deal with this is to create certificates 110 containing the name 112 of the owner of the public key and the public key 108 itself. This certificate is signed with the private key of a well-known certificate signing authority 114, shown in FIG. 6.** The public key of the signing authority is also published in a certificate which is signed by a higher-level signing authority 116. In essence, a hierarchy of certificate authorities has been created.

Id. at 10:18-28 (emphasis added).

D26. Kahn shows how the certificates are used to authenticate digitally signed messages. *Id.* at 22:45-23:45.

D27. Kahn uses digitally signed messages as authorization data, to show that a user has been authorized to purchase or access content. *Id.* at 23:43-45, 24:64-65, 25:39-40, 26:66-67, 27:23-24 (showing use of signed account

information); *id.* at 24:66-25:8, 26:14-22 (“The payment server system then validates the requestor's signature over the contained retrieve-object message 812. If the signature is invalid, an invalid-requestor-signature message is sent to the repository. The payment server validates the account information sent to it and verifies that the account is valid. If the requester is not a valid user of the account, a invalid-user-for-account message is sent to the repository, and the payment server logs the event.”), 27:63-67.

D28. Kahn shows that encrypted communications are used both to transfer data among the repository servers and between the repositories and the end-users. Ex. 1018 (Kahn) at 10:39-48, 10:44-48 (messages can be “sent between systems components via direct connections (e.g., ‘TCP/IP’). The TISPEM library, developed by RSA, is used to provide message privacy and correspondent authentication.”). RSA was a well-known public-key encryption algorithm. While connections would be initiated using public-key techniques, a session key would have been used to encrypt communications during the session. This is because, as Kahn explains, “public key cryptographic algorithms require a substantial amount of computing power,” while “secret key algorithm[s], such as DES, [are] computationally more efficient.” *Id.* at 10:29-38.

D29. Kahn shows that digital objects are encrypted to ensure privacy. *Id.* at 10:29-31. In 1994, a person of ordinary skill would know that encrypting the

content also would make it more difficult for an unauthorized party to redistribute the content. As Kahn explains:

In order to make an object be private in an efficient manner, a combination of public key cryptography and conventional private key cryptography is used. Since public key cryptographic algorithms require a substantial amount of computing power, an object will initially be encrypted with a secret key algorithm, such as DES, which is computationally more efficient. The DES private key will then be encrypted with the public key of the recipient. This encrypted DES key will be sent to the recipient, along with the encrypted object.

Id. at 10:29-38; *see also id.* at 10:39-48, 10:44-48 (messages can be “sent between systems components via direct connections (e.g., ‘TCP/IP’). The TISPEM library, developed by RSA, is used to provide message privacy and correspondent authentication.”).

D30. Kahn shows that “random-value tags” (*i.e.*, nonces) are used to provide communication security during purchase transactions. *Id.* at 24:43-44, 24:52-53, 24:66-25:3. Kahn explains that the random-value tags are used to prevent replay attacks. *Id.* at 25:22-26 (The repository sends back to the user “the original random-value tag or possible a new random-value tag, generated by the repository. This is to avoid play back protection in the event the object identified by the handle is retrieved later.”).

D31. When an author uploads content to be distributed via the repository, the repository will validate the author's digital signature over the object. *Id.* at 2:31-32, 12:60-64, 13:7-13, 13:21-46; *id.* at 16:66-17:44, 19:11-16. If it is invalid, the repository will reject the content. *Id.* at 13:21-46. If it is valid, the repository will store the digital certificate along with other validation information associated with the content. *Id.* at 13:40-46, 1:24-33, 6:65-7:12. Therefore, Kahn shows the repository will validate an object's digital signature before permitting the object to be stored in the repository. *Id.*

D32. Kahn shows that software programs can be distributed to end-users via the repositories. Ex. 1018 (Kahn) at 1:17-25, 5:66-6:5, 6:25-45. Kahn shows that, when a repository provides a user with digital content, the content is signed by the repository. *Id.* at 26:36, 28:18. To validate the digital signature, the end-user uses the repository's certificate, which either is transmitted with the digital work or was previously provided to the end-user. *See id.* at 25:26-28 (requesting system can validate repository's signature). Therefore, an end-user would validate the software's digital signature before installing it.

5. Registration and Purchase Transactions

D33. In the Kahn system, repositories receive requests from end-users to access content. Ex. 1018 (Kahn) at 23:46-50 ("Once an account is established, the user may retrieve an object from the repository . . ."), 24:34-36. Kahn shows that a

user can create an account with a repository, and then purchase and retrieve content. In these processes, digital certificates, digitally signed messages, and other cryptographic techniques are used to facilitate secure transactions. Below, I walk through Kahn's description of these processes.

D34. Kahn explains that “Before a user can retrieve any objects for which payment is required, the user must first establish an account with a payment server system 702 on the network (FIG. 25).” *Id.* at 22:28-30. Kahn shows that when a user creates an account, the following information is used to create a signed message showing the user is an authorized user of the system:

The payment server formats a new-account-response message (728) containing the following:

Account Number

Credit Limit Amount

Time Limit

Categories of Use

List of authorized users (public key certificates plus the certificate chains.)

The requesting system or user's public key certificate chain, which will be used to verify the requestor's identity. Other less efficient methods can also be used, e.g. the payment server could be given sufficient information (the distinguished name) about the user to obtain the certificate chain from another database.

The payment server signs the formatted message and sends it to the user's system. Optionally, the user may be charged a fee for establishing this account and for maintaining it.

Ex. 1018 (Kahn) at 23:24-45 (emphasis added).

D35. Kahn uses this digitally signed message as an authorization message. As Kahn explains that a user will retain this signed message to provide proof of authorization:

The user's system encrypts and stores (730) the received signed message. This account data will be submitted with any activity that may be billed.

Ex. 1018 (Kahn) at 23:43-45.

D36. Whenever a user attempts to purchase content, the user will transmit to the repository a message containing an authorized digital certificate as well as the “account information, previously signed by the payment server.” *Id.* at 25:29-53. The repository will validate the user’s digital certificate and the signed account information message. *Id.* at 25:54-26:22. If the digital certificates and digital signatures are validated, the repository permits the purchase. *Id.* at 26:18-38.

D37. The authorization message contains a “categories of use” field, which specifies the manner in which the message can be used – *e.g.*, it can be used to purchase books and audio, but not video. It also contains conditions on its use, such as a “credit limit” and a “time limit.”

D38. After a user creates an account, the user can search for and retrieve content. After the user finds content to purchase, it obtains a handle, which is a link to the content. *Id.* at 24:25-29 (“An object may be stored in several repositories. Multiple pointers to these repositories may be returned to the requesting system. For each pointer returned by the handle server, the requesting system uses the pointer to attempt to obtain a copy of the digital object 762.”).

D39. After the user finds a valid handle for the content the user wants to purchase, the user requests to initiate a session with a repository:

For retrieval purposes, the requesting system establishes a connection to the repository 766, which takes the form of a small set of transactions.

Id. at 24:34-36.

D40. The repository will evaluate the request to determine if it is authentic. The repository will return a message to the requesting system that includes a “random-value tag”, which is a nonce that will be used to ensure the requesting system is authentic and not replaying messages. *See* ¶ D30, *above*. As Kahn explains:

The repository may examine the calling network address or the requesting system in order to determine if the repository is being inundated with requests from one system. . . .

Normally, however, the repository returns a random-value tag to the requesting system 770. A flag indicating if payment is required to obtain "Terms and Conditions" is included.

Id. at 24:36-46.

D41. Next, the requesting system will request a copy of the terms and conditions for the content. Kahn explains that:

The requesting system needs the object's "Terms and Conditions" before the object can be retrieved. **The requesting system signs and sends the following request-terms-and-conditions message 772 to the repository:**

the object's handle;

the requesting system or user's digital signature over the repository generated random-value tag;

the requesting system or user's public key certificate chain, which will be used to verify the requestor's identity. Other less efficient methods can also be used, e.g. the repository could be given sufficient information (the distinguished name) about the user to obtain the certificate chain from another database. This is needed in the event the repository needs to bill for providing the Terms and Conditions;

a unique tag, assigned by the requesting system;

account information, previously signed by the payment server.

Id. at 24:47-65 (emphasis added).

D42. When the repository receives the message requesting the terms and conditions, the repository will validate the digital signatures and the random-value tag. *Id.* at 24:66-25:3. The repository also validates the signed account information:

The repository verifies the digital signature of the requestor over the repository generated random-value tag 774. If the signature is not correct, the repository sends an invalid-random-value-tag response to the requesting system. The requesting system should log this error.

The repository verifies the payment server's signature over the account information 778. If the signature is not correct, the repository sends an invalid-account-information response to the requesting system. The requesting system should log this error.

Id. at 24:66-25:8.

D43. If everything is valid, the repository will send the terms and conditions back to the user. The terms and conditions can be a list of rights which are available for the user to purchase.

[T]he repository signs the "Terms and Condition" message and sends 792 it to the requesting system, including:

The objectized list of terms/conditions/rights, along with the charge associated with each object and a status flag showing if the term/condition/right is mandatory;

The user-assigned unique key, which was received in the request-terms-and-conditions message;

Either **the original random-value tag or possible a new random-value tag**, generated by the repository. This is to avoid play back protection in the event the object identified by the handle is retrieved later.

Id. at 25:14-26 (emphasis added). Kahn also shows this message includes a nonce (the “random-value tag”) as security protection.

D44. Then the user validates the message by checking the digital signature.

Id. at 25:27-29. If valid, the user selects the parameters of the purchase and returns a signed purchase message.

The user selects the terms and conditions desired 796, including the number of terms (e.g., A user may buy the right to make 5 copies of the object or to perform it ten times). The requesting system uses this information to create the retrieve-object message, including:

the object's handle 798;

the repository generated **random value tag**;

a list of the accepted "Terms and Conditions", including the quantity of each, where applicable;

the user's account information, which was originally signed by the payment server;

the requesting system or user's public key certificate chain, which will be used to verify the requestor's identity. Other less efficient methods can also be used, e.g. the repository could be given sufficient information (the distinguished name) about the user to

obtain the certificate chain from another database. the domain name and the port number which are used by the requesting system to receive the object.

limitations, if any, on the object by the requesting system (e.g. maximum object size it can receive).

The entire message is signed by the requestor. This is similar to signing a credit card slip.

Id. at 25:30-53 (emphasis added). The signed purchase message includes the requested terms and conditions, the user's digital certificate, the user's signed account information, and the nonce. The message is digitally signed by the user.

D45. The repository then initiates a connection with a payment server. The repository provides its digital certificate, and sends the request and other data to the payment server. *Id.* at 25:59-26:9. If the payment server determines the transaction is authorized, it informs the repository. *Id.* at 26:10-30. The repository then sends the object and the purchased usage rights to the user:

Otherwise an account-has-been-debited message is signed by the payment server and sent to the repository 818.

The repository connects to the address/port specified in the request, and it transmits 820 to the requesting system:

the object's handle;

the total amount debited from the account;

the object, signed by the repository;

portions of the relevant terms and conditions, if appropriate.

Id. at 26:31-38.

D46. As I explained above, each of the messages exchanged between the components of the system, digital certificates and digital signatures are used to authenticate the identity of the parties to the message and ensure their integrity. *See* ¶¶ D22-D23, D28, *above*. Signed messages are used to provide evidence of authorization. *See* ¶ D27, *above*. Many of the messages include random-value tags, which are nonces used to prevent replay attacks. The communications are encrypted to ensure privacy. *See* ¶ D30, *above*.

D47. Although Kahn discloses that a user can render or perform digital content in accordance with the terms and conditions of use, Kahn does not explicitly describe the devices or software used to render content.

D48. A person of ordinary skill in the art would have understood that Kahn focused on describing server-side systems for securely distributing content and left the client-side security to those implementing the system.

D49. When implementing a system like Kahn, a person of ordinary skill in the art would have incorporated client-side techniques to ensure content would continue to be protected after being received by users. A person of ordinary skill would not have ignored the user-side of content protection; instead, such a person would have looked to known solutions to implement user-side protections.

B. Overview of Linn (Ex. 1058)

D50. In “Copyright and Information Services in the Context of the National Research and Education Network,” R.J. Linn describes a set of mechanisms for protecting electronic content. Ex. 1058 (Linn) at 14-15; *id.* at 13 (“New protective services can be created for information dissemination . . .”).

D51. Linn explains its techniques can be used to protect any type of content, including video, audio, and books. *Id.* at 20 (“5. The techniques described in this paper are equally applicable to output media other than video displays and printers. Thus . . . , ‘rendering software’ [] present[s] the content of the information in a human interpretable form: video, audio, printed text or some combination thereof.”). Linn explains that all of the mechanisms discussed in the paper “are applicable to **any information** distributed over a computer network... .” Ex. 1058 (Linn) at 17 (emphasis added).

D52. In the Linn system, content is distributed in a signed envelope that includes the content and basic usage rights information on it. Ex. 1058 (Linn) at 14-15. **The digitally signed envelope joins the content and associated usage rights.**

D53. Linn describes an envelope and the content associated with it as an “information object.” Ex. 1058 (Linn) at 14 (“For discussion purposes, we consider a body of material (information) as an ‘object’ with certain components

and attributes. . . . For simplicity, however, we describe an object as an envelope and its contents. The information on the envelope is visible and the contents hidden and sealed with a digital signature.”).

D54. Linn explains that when the content and envelope are joined, the content is “hidden.” Ex. 1058 (Linn) at 14. Linn does not explicitly state that the content would be encrypted. However, a person of ordinary skill in the art would find encrypting the content in the envelope to be obvious. I discuss this in more detail below, with respect to the combination of Linn and Kahn.

D55. Linn explains that its methods rely on special software called “rendering software.” The rendering software is used to create, exchange, and render content. For example:

- An author or content distributor (*e.g.*, an “information service”) will use the rendering software to package content into information objects. Ex. 1058 (Linn) at 15, 16; *see id.* at 14 (“Dissemination may be by an author, original publisher, information service, or library (hereafter called an authorized distribution source).”).
- Content distributors will use the rendering software to distribute content to end users. As Linn explains: “[c]opies of objects are exchanged using the rendering software.” Ex. 1058 (Linn) at 15.
- An end user will use rendering software to process, decrypt, and display the content in each object. Ex. 1058 (Linn) at 13, 14, 15. In fact, in the Linn system, the **only** way a user can render content in an object is by using the rendering software. *Id.* at 16 (“the rendering software is required to display

or print an object.”); *id.* (“an object is stored in a form which may not be displayed or printed without the rendering software unless it is extracted from within its envelope.”).

D56. Linn explains that it is the rendering software, and not the content, that enforces the usage rights on an information object. As Linn explains: “The means to assure authenticity of materials and achieve protection from deliberate abuse by end users is to **implement the required protection mechanisms in computer systems as part of the application programs which deliver services to users.**” *Id.* at 13; *id.* at 15 (“Active Protection Mechanisms. **Two active-mechanisms implemented in the rendering software** will achieve the requirements for protection outlined in the previous section.” (italics in original)); *id.* at 16. Linn shows that the rendering software enforces the usage rights both when an object is received and before rendering the content in an object. Ex. 1058 (Linn) at 15-16 (“If verification failure is detected ***when an object is being obtained*** from an information service, its retransmission should be requested. (This might occur if data were lost or corrupted.) If a failure is detected ***when displaying or printing an object***, further processing should be inhibited. This might indicate a bootleg copy . . .” (emphasis added)).

1. Information Objects

D57. Linn explains that the content in the information object is hidden using cryptographic techniques, and it is “difficult, if not impossible” for a user to

extract the content from the object without the rendering software. *Id.* at 13; *id.* at 14, 15-16.

D58. An example of the envelope is shown below:

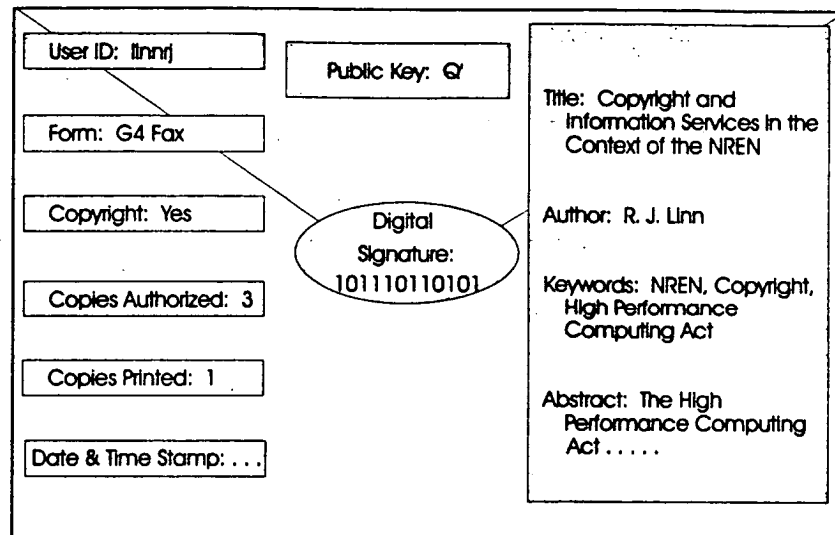


Figure 2: An Information Object -- Envelope plus Content

Ex. 1058 (Linn) at 15; *id.* at 14 (To discourage plagiarism, excising parts of the text and other unauthorized uses of the information, an object could be put in a ‘sealed envelope’ and distributed in one of several forms which are not easily read and modified by humans.”).

D59. In this example, the fields of the envelope are the following: User identification string, form of the content, a flag indicating whether or not the content is copyrighted, number of printed copies authorized, date and time envelope was sealed, the public key of the authority that signed the envelope (*e.g.*,

author or publisher), the title-author-keywords-abstract of the content, and a digital signature of all of these fields together with the content.

D60. Linn describes many of these fields in the following passage:

Limited redistribution: Identifying the holder of the copy on the envelope (e.g., user identification) and **a copy counter can be employed to limit electronic redistribution.** The **user identification** and initial value of the copy counter stored on the envelope are established when a copy is obtained from an authorized distribution source. The number of printed copies allowed is a function of the fee paid. **The copy counter is used to restrict the number of copies rendered on a printer.**

Ex. 1058 (Linn) at 16 (emphasis added); *id.* at 15; *id.* at 16 (“Materials may be displayed on a video display an unlimited number of times by the user identified on the envelope. Other users are prohibited from displaying an object with the ‘copyright’ attribute.”).

D61. Linn explains that an author can limit redistribution of content using the user identification field. Ex. 1058 (Linn) at 16 (“Limited redistribution: Identifying the holder of the copy on the envelope (e.g.; user identification) and a copy counter can be employed to limit electronic redistribution. . . .”). Similarly, if an author or distributor does not want to limit redistribution, unlimited redistribution can be specified using the fields on the envelope:

[U]nlimited rendering and redistribution is permitted if an authorized distribution source omits the ‘copyright’ attribute on the envelope, or enters ‘unrestricted’ in either the user identification or copies authorized fields.

Ex. 1058 (Linn) at 16; *id.* at 15 (“an authorized distribution source []may be an individual if there is no fee for use”).

D62. A person of ordinary skill in the art would have found it obvious to add additional fields to the envelope to specify additional rights or restrictions on use. For example, a field could be added to limit the number of times the content was electronically rendered. Or a field could be added with an expiration date, allowing for rented content.

D63. The fields on the envelope are unencrypted, and hence, visible and searchable. Ex. 1058 (Linn) at 14 (“Visibility of information on the envelope has other obvious advantages related to search and retrieval of information stored in digital libraries, but they are outside the scope of this paper.”). As Linn explains, the rendering software is intended to be integrated with other content distribution technologies, such as digital libraries. *Id.*

D64. Although Linn states that the envelope contains the public key of the authority that signed the envelope, a person of ordinary skill in the art would have understood this to mean a digital certificate containing the public key of that authority. Linn does not explicitly provide details about this process, because the

process of distributing public key information was well-known by 1994. I describe some of these well-known techniques in ¶¶ A134-A160, *above* and ¶¶ D22-D30, *above*.

D65. Linn shows that the envelope is a distinct data structure that is stored in clear text, and it contains fields that describe the content and specify the usage rights. Ex. 1058 (Linn) at 14-15. The content file can be of any data format (*e.g.*, an SGML document or encoded using “G4Fax”). *Id.* at 14, 15.

D66. As Figure 2 shows, an “information object” comprises two different parts, which could exist in two separate files: an “envelope plus content.” *Id.* at 15. It would have been a routine implementation choice to deploy a system based on Linn which kept envelope together with the content in a single file structure or to separate these two elements of the information object into discrete files and logically associate them. For example, the envelope could be stored in a database to facilitate search functionality, while the content stored in memory. Or, the envelope and content could be stored together in the same memory structure.

D67. I note that the claims of the ’072 patent specify that content is stored in a separate file from the usage rights. The ’072 patent shows that the content file and the description data (which contains the usage rights) are separate files but are part of a contiguous block of data. *See* Ex. 1003 (072) at 9:60-63 (“The approach for representing digital works by separating description data from content assumes

that parts of a file are contiguous but takes no position on the actual representation of content.”). Under the same rationale, Linn’s storage of content and the envelope (containing usage information) in separate files that are part of the same block of data are stored in “separate files,” within the meaning of the claims.

D68. I believe it also would have been obvious to the person of ordinary skill that the envelope could be stored a separate, distinct file from the content without affecting how the Linn scheme would function. For example, Kahn shows that usage information can be stored in an external properties record. *See* ¶ D10, *above*. Configuring rendering software to store an envelope and content separately, as Kahn shows with the properties record, would have been a routine engineering task. For example, Linn shows that the usage information on the envelope is in clear text to facilitate integration with other systems, such as a digital library search system.

D69. It would have been obvious to store the content (which a person of ordinary skill in the art would have encrypted) in one physical memory location, while storing the envelope in another physical memory location to allow it to be more easily searched. The digital signature over the combination of the envelope and content serves to logically bind the two to each other because the signature can only be validated if one has both parts (the content and envelope). *See* ¶ A97, *above*.

2. Rendering Software

D70. As I explained above (¶ D55), Linn explains the rendering software is used for all aspects of an information object's lifecycle.

For our purposes we assume:

- an object is processed by standardized software (hereafter called rendering software);
- **creating an original information object, file transfer over a network, and rendering of the information on a video display or printer are built-in functions of the rendering software;**
- the rendering software is
 - inexpensive or free because it is in the interest of the public, and of publishers and authors to protect their intellectual property, and
 - widely available; e.g., distributed by publishers, information service providers, computer manufacturers;
- copies of objects are exchanged using the rendering software-a copy is obtained from an authorized distribution source (may be an individual if there is no fee for use); and
- the structure and exchange formats of objects are standardized (either de facto or de jure). [meaning by an official industry standard or an informal standard]

Ex. 1058 (Linn) at 15 (emphasis added).

D71. Linn explains that end users access the content in an object by using the rendering software. The rendering software can render the content on a video

display, a printer, or another device as appropriate. Ex. 1058 (Linn) at 13-14 (“ . . . Processing software employed by a user should display or print the materials in an appropriate form given the constraints of the user’s video display and/or printer.”); *id.* at 15 (“rendering of the information on a video display or printer [is a] built-in function[] of the rendering software . . .”).

D72. Linn shows that the rendering software runs on a computer system. Ex. 1058 (Linn) at 13 (“implement required mechanisms in computer systems as part of the application programs . . .”), 14 (“A set of mechanisms may be combined . . .”), 15 (“an object is processed by standardized software [] called *rendering software*”).

D73. Linn explains that the rendering software and the cryptographic techniques used to seal the envelope prevent users from removing content from the object. Linn explains that the only way content can be accessed is with the rendering software. Ex. 1058 (Linn) at 16 (“Note that all the forms described above prevent easy redistribution by simply making a copy and mailing or printing it with utility software because **the rendering software is required to display or print an object**. These forms also inhibit using a simple editor to ‘cut and paste’ text into another document because no form is human readable, and **direct user access into the contents of the envelope is not allowed by the rendering**

software.”). This allows the rendering software to control whether and how users access the content.

D74. In the Linn system, the **only** way content can be obtained and exchanged is by using the rendering software. *Id.* at 15. If an information object is copied or transferred to a user via email or another medium, the recipient’s rendering software will detect that copy is unauthorized, will refuse to render it, and may destroy it. *Id.* at 16 (“the object could be destroyed by the rendering software when verification fails”), 13.

D75. Linn explains an information object may be distributed only by “an authorized distribution source,” (*id.* at 15), and that unauthorized copies are useless. *Id.* at 16 (“[S]ending a copy of an object via electronic mail, redistribution by a bulletin board and other simple copying mechanisms will not update the contents of the envelope which contains the date and timestamp . . . [and] the rendering software considers the copy to be unauthorized. Consequently, the information contained in the envelope will not be presented to a user by the rendering software and unauthorized copies are useless.”).

D76. Linn explains the rendering software restricts rendering rights based on the identity of the user. *Id.* at 13 (“Limited redistribution: Publishers want to control distribution to those who have paid a fee for the use of copyrighted materials. Mechanisms should be implemented to restrict the number of copies

printed to those paid for and to the individual who paid for them.”); *id.* at 16 (“If a failure is detected when displaying or printing an object, further processing should be inhibited. This might indicate a bootleg copy, or **a mismatch of user identification with that on the envelope . . .**”).

D77. Linn shows that an end-user can request content from a distribution source. *See also* Ex. 1058 (Linn) at 14 (“Dissemination may be by an author, original publisher, information service, or library (hereafter called an authorized distribution source). . . [A]n individual **could ask for and get a copy of a paper or article** as easily as he or she can reproduce it on a copier in a library (and at a comparable price). Remuneration by an individual patron could be **at the time the material was obtained**, if there was a fee.”).

C. Tamper-Proof Hardware

1. Overview of Dyad

D78. I introduced the Dyad publication at paragraphs A74-A75 and A131-A133, *above*.

D79. Dyad explains a technique where a tamper-resistant secure coprocessor “can be directly inserted in standard workstations or PC-style computers.” Ex. 1057 at 121. As the paper explains, its techniques provide “easily implementable solutions” to security problems (*id.*), and is “cost effective,” (*id.* at 124). *See id.* at 127, 133, 143.

D80. Dyad also explains how the coprocessor's secure cryptographic functionality can be used to provide various security services to a system. *Id.* at 124, 127 ("Applications" section), 140-41; *see id.* at 127-36.

D81. These include mechanisms for (1) checking the integrity of the host computer using cryptographic checksums, (2) using digital certificates to certify copies of software, (3) using digital certificates to provide copy protection that prevents a purchased and installed copy of a software product from being copied onto and used on a different computer, and (4) creating "tokens" that can be used to transfer the right to execute a program. Ex. 1057 (Dyad) at 121, 127-132.

D82. Dyad also explains that the secure coprocessors improve the security of systems by providing for a "tamper-proof audit trail[] and accounting logs." Ex. 1057 at 123, 130-31, 139-41.

D83. As I explained above, the Dyad secure processor builds on the software protection techniques described in ABYSS, and modifies them to use public-key encryption techniques. *See ¶¶ A131-A133, above.*

D84. Dyad also suggests that its secure coprocessor can be used to validate the integrity of system software during installation, at boot time, and before execution of the software. *Id.* at 141, 138 ("[A] secure coprocessor verifies the system software . . . by checking the software's signature against known values."), 140 ("***Authentication, key management, fingerprints, and encryption crucially***

protect the integrity of the secure coprocessor software and the secrecy of private data, including the secure coprocessor kernel itself.”), 127-130.

2. Overview of Shear

D85. Shear describes a system for ensuring that content providers are paid for end-users’ use of content. Ex. 1055 (Shear) at Abstract, 3:49-59.

D86. Shear’s system focuses primarily on an end-user device. The end-user device locally stores digital content, decrypts it, renders it, meters the amount of usage, and bills the user. However, Shear also discusses several embodiments that include a remote central service that provides digital content via a modem or other communications link.

D87. Shear’s system includes four main components: a hardware device plugged into a host computer, a storage medium containing a database of digital content, a decoder (decryption) unit/biller block, and a database access program in the host computer.

D88. The “hardware device” is a “tamper proof hardware module that prevents unauthorized and/or unmetered use of a protected information [data]base.” *Id.* at 3:63-66, Abstract. The hardware device is plugged into the host computer. *Id.* at Abstract.

D89. A storage medium such as a CD-ROM contains a database of encrypted digital content, such as magazines. Ex. 1055 (Shear) at 4:10-20, 5:65-68, 6:38-42, 6:59-62, 9:34-44.

D90. A “database access program resident in the host computer” permits the user to formulate a database access request. *Id.* at 5:3-10, 11:11-38, 14:37-43, 16:57-62. The database access program also “controls the host computer to display and/or print the decrypted information.” *Id.* at 16:57-62.

D91. A tamper-proof “decrypting device and associated controller” in the hardware device or on the host computer decrypts selected content, meters usage, and allows billing the user according to the amount he has used the database. Ex. 1055 (Shear) at Abstract, 3:49-59, 5:11-32, 6:38-47, 9:13-33.

D92. Shear’s system allows the database owner or end-users to specify various restrictions on content usage. *Id.* at 16:57-62, 18:10-12, 18:53-56, 19:7-11. The restrictions can include preventing the user from “doing anything other than displaying (and/or printing) the decrypted data,” restrictions on exceeding “a predetermined maximum (and/or minimum) usage limit,” and limits on duration, cost, and type of information which can be decrypted. *Id.* Shear’s system also allows charging the user different rates for browsing, copying, modifying, printing, or other types of usage. *Id.* at 7:6-15.

D93. Shear's system includes numerous security features. The security features include a "tamper-proof decrypting device," encryption of content stored in the hardware device, "mechanical and electronic safeguards," and security codes. *Id.* at Abstract, 3:63-66, 5:11-29, 8:6-12, 12:57-66. Shear shows that these security functions are performed by both the hardware device and the computer it is attached to. Shear also states that these functions could be integrated into a dedicated independent hardware unit ("browsing workstation") (*id.* at 20:33-63), or they could be integrated into a special software program loaded into a host computer (*id.* at 21:15-27).

D94. Shear also explains that the database containing digital content might be located remotely, at a central facility. Ex. 1055 (Shear) at 19:27-64. Shear explains that its invention can be used advantageously with digital libraries. *Id.* at 6:38-62, 7:34-8:5, 14:51-57.

D. Kahn and Linn Suggest a Combined Server/Client System

D95. A person of ordinary skill in the art would have found it obvious to combine the Kahn repository system for distributing content with the rendering software and information objects described in Linn. One reason is that the person would have recognized that Kahn and Linn are directed toward complementary aspects of the same problem: the secure distribution of content. Another is that the combined system would improve the security and effectiveness of both systems.

D96. Both Kahn and Linn describe techniques for securely distributing digital content with terms and conditions: Kahn is directed toward the servers that distribute content and usage information to users, and to the process for a user to download or retrieve information. Kahn does not, however, describe how the terms and conditions of use are enforced after content is transferred to a user. *See* § VII.A, *above*.

D97. Linn describes sealed “information objects” that contain usage information and shows using “rendering software” that permits access to the content but only in accordance with the terms and conditions of use that are specified within the objects (*i.e.*, in an “envelope” field). *See* § VII.B, *above*.

D98. Although Linn explains that content is downloaded from an information service in its model, Linn does not provide details about the server-side of the process or the interactions of client devices with the server.

D99. The two systems have many architectural similarities. Both Kahn and Linn describe distributing content in a manner that allows authors to specify restrictions on usage of content. Both also show that end-users can request content from a server – a repository in Kahn and an information service in Linn. Both show use of public-key encryption techniques to ensure the integrity and authenticity of content. Both show that content can be any type of digital content. *See* §§ VII.A & B, *above*.

D100. A person of ordinary skill in the art considering how to implement the Kahn system would have naturally considered how to design and implement a user-side enforcement mechanism in end-user devices that would receive and use content. A person of ordinary skill would not simply ignore the client-side of the distribution process. Instead, that person would have looked to known techniques for client-side enforcement, and would integrate a client-side solution into the Kahn system.

D101. The Linn rendering software would be an obvious choice for a client-side protection technique to combine with the Kahn servers. In addition to the technical similarities between the two approaches, the Linn rendering software-based approach for protecting content is consistent with the Kahn system.

D102. For example, Kahn explains that authors use a custom software application to upload content to the repositories. Ex. 1018 (Kahn) at Figs. 10, 14 & 18; *id.* at 20:24-22:25. It would have been obvious to extend that custom software-based approach to the end-user devices that will “consume” the content. In fact, the client devices in Linn that run its rendering software would be a natural extension of the Kahn system.

D103. Combining the Linn client devices with rendering software with the Kahn repositories would require only routine effort to implement. In fact, one could adjust the configuration of the combined system to meet any unique

engineering goals or to fit the specific requirements of any particular implementation. For example, a person of ordinary skill could alter the cryptographic techniques used to provide additional (or less) security for the content, to improve the system's performance, or to make the system easier to use.

D104. In the combined system, the Kahn repositories would be used to distribute content, and the Linn rendering software would be used to download content from the repositories and to render content in accordance with specified terms and conditions.

D105. A person of ordinary skill would have found it logical to provide users with rendering software that performs the user-side of the Kahn protocols. Each component of the combined system would function as it was described in Kahn or in Linn. Combining these two systems would be a routine engineering task for a person of ordinary skill in the art, requiring only minor modifications to each system to integrate them.

D106. For example, a person of ordinary skill in the art could have configured the Linn rendering software to interface with the Kahn repositories using the user registration and transaction protocols described in Kahn. Ex. 1018 (Kahn) at 22:28-23:45.

D107. Linn shows that the rendering software is used to download content from an authorized distribution source or an "information service," but does not

identify a protocol for such transactions. Ex. 1058 (Linn) at 14. The Linn rendering software could be configured to allow a user to obtain the required digital certificates and to register with the repository and create an account as described in Kahn. See ¶¶ D33-D41, *above*. The Linn rendering software also could be configured to implement the Kahn transaction protocols for purchasing and downloading content. See ¶¶ D33-D49, *above*. Merging these elements would be logical because Kahn describes the protocol but not the specific software application used by the end-user, while Linn explains that content is exchanged using the rendering software, but does not provide a protocol for such exchange.

D108. Configuring the rendering software to obtain the digital certificates and account information described in Kahn also would improve the operation of the Linn enforcement mechanisms. Linn explains that the envelope can contain a “user ID” and that the rendering software ensures that the “user ID” on the envelope matches the user ID of the person using the rendering software. Ex. 1058 (Linn) at 15-16. It would have been obvious to use the digital certificates and data in the digitally signed account information (*e.g.*, account number) described in Kahn to authenticate users of the rendering software and to verify that an information object is associated with the user’s ID. and configure the rendering software accordingly

D109. In the combined system, the rendering software would need to be distributed to end-users wishing the Kahn repositories. It would have been obvious to have the entity operating the repository (*e.g.*, the publisher or provider) to provide the rendering software to end-users. This is suggested by Linn, who explains that rendering software should be widely available and can be distributed by “publishers [and] information service providers.” Ex. 1058 (Linn) at 15.

D110. A person of ordinary skill would find it obvious to have the repository distribute the rendering software with a digital signature and digital certificate to enable users to validate the authenticity and integrity of the rendering software itself before installing it. This was a common practice in 1994, especially for software that would handle financial or purchase transactions (for example, Kahn describes transmitting a credit card number to a repository and making charges against it, Ex. 1018 (Kahn) at 22:30-34 & 22:46-48). Using digital signatures and certificates to validate the integrity and authenticity of the rendering software is simply a logical extension of the techniques described in both Linn and Kahn, each of which shows use of digital signatures to validate the authenticity and integrity of content before that content can be stored or used. Ex. 1018 (Kahn) at 25:26-28, 26:36, 28:18 (when an end-user receives content from the repository, the end-user validates the repository’s digital signature on the content using repository’s digital certificate); Ex. 1058 (Linn) at 15-16.

D111. Kahn shows that when a user purchases digital content, both the content and terms and conditions are transferred to the user. Ex. 1018 (Kahn) at 26:31-38. Kahn does not expressly identify how the terms and conditions are enforced on the user-side.

D112. To enable the Linn rendering software to enforce the terms and conditions of use, the Kahn repositories could be configured to package digital content into the Linn information objects, and to include the terms and conditions of use as data fields on the envelope. A person of ordinary skill could have readily integrated the two schemes by adding fields within the Linn envelope to accommodate the various terms and conditions described in Kahn. Linn explains that it is describing only an example of a possible envelope (Ex. 1058 (Linn) at 14), and a person of ordinary skill easily could have expanded the fields on the envelope to accommodate the additional usage rights described in Kahn. *See* ¶¶ D6-D8, *above* (use, perform, modify, transfer); Ex. 1018 (Kahn) at 7:16-22. Doing so would require only routine and predictable effort.

D113. Linn shows envelopes and content can be implemented as a formatted file using a markup language such as SGML. Ex. 1058 (Linn) at 14 (“an object could be put in a ‘sealed envelope’ and distributed in [a] form[] which [is] not easily read and modified by humans . . . [such as] SGML”).

D114. In 1994, markup languages such as SGML were well-known. SGML is extensible, meaning that the format of SGML files can easily be modified, and the dictionary of tags used in a particular SGML scheme is simple to modify. *See* Ex. 1062 (HTMLv2.0) at 2 (“The SGML declaration determines the lexicon of the grammar. It specifies the document character set, which determines a character repertoire that contains all characters that occur in all text entities in the document, and the code positions associated with those characters.”), 10-12. Using SGML to represent the terms and conditions or usage fields on the Linn envelope would have been an obvious way to accommodate Kahn’s types of usage rights and would add flexibility to the combined system. This configuration is suggested by Linn itself.

D115. As another example of how SGML improves the combined system, it would simplify the process of distributing the compound objects, described in Kahn: “[a] digital object may contain other digital objects.” Ex. 1018 at 6:17-20. One common example of such a compound object might a book containing images, or a magazine or newspaper containing multiple articles and images. Ex. 1018 at 1:15-21. Similarly, Kahn explains a compound object can be “a concatenated sequence of digital objects,” where each object includes a properties record that specifies where each object starts and ends, and specifies the terms and conditions for that particular object. Ex. 1018 at 6:8-16, 6:39-43. A convenient way to implement such functionality would have been by integrating the objects into an

SGML document, and by using tags to identify and arrange the distinct objects that compose the document. *See* Ex. 1062 (HTMLv2.0) at 2, 10-12. By using tags to represent the terms and conditions, each component could retain its own terms and conditions.

D116. When integrating the information objects into the Kahn system, encrypting the content in the information object would make it more difficult for end-users to extract the content without using the rendering software. A content encryption technique is described in Kahn, which explains that content is “encrypted with a secret key algorithm, such as DES” and then the “DES [secret] key will [] be encrypted with the public key of the recipient.” Ex. 1018 (Kahn) at 10:29-38. Kahn explains that the “encrypted DES key will be sent to the recipient, along with the encrypted object,” which in the combined system could be accomplished by embedding it into the envelope in the information object. *Id.* This technique would ensure that only the end-user (who is the only entity possessing the private key that can decrypt the DES key) could access the content.

D117. A person of ordinary skill could readily integrate this known technique with the Linn envelopes to ensure that only the user with the corresponding private key (*i.e.*, the user identified in the digital certificate) could decrypt the symmetric key and access the content. After the user acquired an object from the repository, the Linn software would be used to render content, and

enforce those terms and conditions. Just as Linn describes, the rendering software would check the usage information on the envelope before rendering or transferring content. See ¶¶ D55-D62, D76, *above*.

D118. A person of ordinary skill in the art would have found it to be logical to combine these two complementary systems. To summarize the discussion above (in ¶¶ D95 to D117), the Linn rendering software could be configured to follow Kahn's registration protocol to obtain an authorization message from the payment server. Once obtained, the user can select desired content, and then using Kahn's purchase transaction protocol, purchase the content and the terms and conditions set by the author. The repository server will determine package the content and terms and conditions into an SGML envelope as described in Linn, which allows for more flexible and more granular terms and conditions to be specified (such as those in Kahn). The Kahn repository server would then send the envelope with content as an information object to the Linn rendering software. The information object would be stored on the user's computer until the user requested to use it, at which point the Linn rendering software would process the request, check it against the terms and conditions in the envelope, and determine whether to proceed with rendering.

E. Kahn in View of Linn Suggests a System that Will Use “Usage Rights”, “Repositories, and “Authorization Objects”

1. A Kahn / Linn System Would Use “Usage Rights”

D119. Kahn shows that terms and conditions can specify the particular rights granted to a user, such as the right to “perform,” “modify,” “transmit,” and “enhance” the content. Ex. 1018 (Kahn) at 7:17-22, 2:40-46, 3:4-13 (each digital object in a repository is associated with terms and conditions of use). Each of these rights corresponds to a particular action that can be taken with respect to the content.

D120. Linn shows an envelope includes data fields for specifying usage information, such as an indicator as to whether physical copies are permitted, and a copyright attribute that indicates whether a user can further distribute the content. Ex. 1058 (Linn) at 15-16.

D121. As I explained above (¶¶ D55-D62), a person of ordinary skill integrating the two systems would have found it to be an obvious implementation choice to augment the Linn envelope by adding fields to store the Kahn “terms and conditions.”

D122. In the combined system, the fields on the Linn envelope would be expanded to accommodate the additional rights provided in Kahn. See ¶¶ D55-D62, *above*. As Linn points out, additional fields can be added to its envelope (Ex. 1058 (Linn) at 14 (fields are exemplary)), and a person of skill could have used the

added fields to the Linn envelope to specify the various manners of use identified in Kahn, such as the rights to “obtain, use, modify, transmit, perform, and enhance” the content. Ex. 1018 at 7:17-22; Ex. 1058 (Linn) at 15-16. The expanded envelope would contain fields used to define specific actions or “manners of using or distributing” the content such as the right to render, physically print, transfer (redistribute), or modify content. See Ex. 1001 at 18:35-19:53 (Possible rights include “play,” “copy,” “transfer,” “edit,” and “install”).

D123. The combined system would those have a right element or a “manner of use” within the meaning of the claims. See ¶¶ B32, B45, B59, *above*.

D124. The expanded Linn envelope would also have fields used to specify conditions. Kahn shows that terms and conditions can limit the right to render content or the right to copy content to a finite or defined number of times: “The user selects the terms and conditions desired [], including the number of terms (e.g., A user may buy the right to make **5 copies** of the object or to perform it **ten times**).” Ex. 1018 (Kahn) at 25:29-31.

D125. Linn shows that the envelope can contains several fields for placing conditions on use of the content including, *e.g.*, a “user identification” or “user ID” field, which can be used to restrict distribution to a particular user, and a “copy counter,” which tracks how many copies of the content have been printed and limits the number of copies that can be printed. *Id.* at 15-16.

D126. Therefore, the combined system would have a conditions element that limits the specified “manner of use” within the meaning of the claims. *See* ¶¶ B33, B46, *above*.

D127. In the Stefik system, usage rights are permanently attached to a particular copy of the content. As I explained above, the Linn content and the envelope are joined and sealed with a digital signature – this makes a permanent attachment of the content to its envelope. *See* ¶¶ A97, D69, *above*. Kahn also requires the terms and conditions to be attached to content when it is stored in a repository server. Ex. 1018 (Kahn) at 11:16-22 (“the repository contains a copy of the object plus identification of certain simple terms and conditions for obtaining a copy of the object and using it”).

D128. A person of ordinary skill would have considered it an obvious implementation choice to configure the combined Kahn/Linn system to distribute content packaged in the Linn information objects. The Linn information object combines the digital content with an envelope that has data fields for specifying usage rights and conditions. Ex. 1058 (Linn) at 15. The envelope is sealed by the author using a digital signature, and consequently, the content cannot be rendered if the content or envelope is altered. Ex. 1058 at 15-16. The Linn envelope also contains the decryption key for the content, and consequently, the content cannot be used without the envelope. The usage rights in the envelope in a combined

Kahn/Linn system would be enforced by the Kahn repositories, and by the rendering software on an end-user's device. Ex. 1018 (Kahn) at 3:9-13, 3:27-31, 8:25-29, 26:7-30; Ex. 1058 (Linn) at 13-16. Thus, in the combined Kahn/Linn system, the "usage rights" would be enforced by "repositories."

D129. A person of ordinary skill would also have found it obvious to represent each right and condition using statements in a markup language such as SGML – this is explicitly suggested in Linn. Ex. 1058 (Linn) at 14 ("an object could be . . . distributed in [a] form[] . . . [such as] SGML"). Thus, each usage right defined in the Linn envelope field would be a statement in a usage rights language (*i.e.*, the SGML schema would define a fixed number of values reflecting the "language"). See ¶¶ D114-D115, *above*; Ex. 1062 (HTMLv2.0) at 2, 10-12.

2. A Kahn / Linn System Would Use "Repositories"

D130. In the combined Kahn/Linn system, both the Kahn repository servers and the Linn computer running the rendering software would be a "repository" within the meaning of the Stefik patents.

a. The Kahn Repository Servers

D131. The Kahn system uses "repository servers" that store content "using a storage technique which renders the digital objects secure against unauthorized access." See, *e.g.*, Ex. 1018 (Kahn) at 2:17-21; *id.* at 3:30-32, 4:46-49 (digital objects are stored "in a secure manner that both restricts access and allows for later

verification that the deposit[ed] [content] has not been altered”), 8:27-29 (“The repository 36 holds copies of digital objects in a secure and verifiable manner and controls access to the objects.”).

D132. In the Kahn systems, before being able to receive and use content, each user must obtain a digital certificate, register with a payment server, and obtain an account number and a signed account certificate. *See* ¶¶ D33-D37, *above*; Ex. 1018 (Kahn) at 22:45-67, 23:21-45. In the Kahn systems, these digital certificates are used to identify authorized users, and content will not be provided to a user unless the user is authorized. *See* ¶¶ D37-D46, *above*; Ex. 1018 (Kahn) at 22:46-48; *id.* at 22:45-23:45.

D133. Therefore, the Kahn repository servers possess physical integrity. *See, e.g.*, Ex. 1001 (859) at 11:67-12:5 (physical integrity mean the repository “never allows non-trusted systems to access the digital works directly.”), 12:16-18 (“repositories prevent access to the raw data by general devices and can test explicit rights and conditions before copying or otherwise granting access”).

D134. The Kahn repository servers use public-key technologies such as digital certificates, digital signatures, hash functions, and privacy enhanced mail to provide system integrity, privacy, authentication, and secure communications. *See* ¶¶ D22-D30, *above*; Ex. 1018 (Kahn) at 5:6-8, 9:41-43, 10:10-38.

D135. In the Kahn system, content is encrypted. *See* ¶¶ D22-D23, D28, *above*; *id.* at 9:37-43 (“As objects and other information are transmitted between the various system components, the privacy of the information must be ensured. The system uses available public key and digital signature technology to handle privacy and authentication in the system as follows.”).

D136. In the Kahn system, encrypted communications are used both to transfer data among the repository servers and between the repositories and the end-users. *Id.* at 10:39-48, 10:44-48 (messages can be “sent between systems components via direct connections (e.g., ‘TCP/IP’). The TISPEM library, developed by RSA, is used to provide message privacy and correspondent authentication.”). As Kahn explains, when two components of the system communicate directly, the communications are encrypted using conventional processes, such as those in a standard encryption library. RSA was a well-known public-key encryption algorithm, and an encrypted session would be established by using digital certificates and public-key encryption to exchange a session key. *See* ¶¶ A138-A153, *above*.

D137. Kahn explains that digital certificates and digitally signed messages are used for authentication and to identify authorized users. *See* ¶¶ D23-D28, *above*; *see, e.g.*, Ex. 1018 (Kahn) at 5:5-8, 6:65-7:4, 10:10-17; *id.* at 10:18-28, 22:45-23:45.

D138. In the Kahn system, the user's digital certificates are validated when the user attempts to initiate communications and purchase content. *Id.* at 24:34-25:59, 26:26-38, 10:18-38.

D139. Kahn shows that random numbers (*i.e.*, nonces) are used to provide communication security during purchase transactions. *See* ¶¶ D30, D39-D46, *above*; *e.g.*, Ex. 1018 (Kahn) at 24:43-44, 24:52-53, 24:66-25:3.

D140. Therefore, the Kahn repository servers possess communications integrity. *See, e.g.*, Ex. 1001 (859) at 12:24-33 (repositories can use "security measures involving encryption, exchange of digital certificates, and nonces").

D141. The repository servers in the Kahn systems are used to distribute content, including software. *See* ¶¶ D31-D32, *above*. When an author uploads content to the repository, the repository will validate the author's digital over the object. *See* ¶¶ D31-D32, *above*; *see* Ex. 1018 (Kahn) at 1:24-33, 2:31-32, 6:65-7:12, 12:60-64, 13:7-13, 13:21-46; *id.* at 16:66-17:44, 19:11-16. Thus, the repository will validate an object's digital signature before permitting the object to be stored. *Id.*

D142. Kahn does not explain how the software that provides the server-side repository functionality is distributed. A person of ordinary skill in the art would have found it obvious to distribute that software with a digital signature and digital certificate, and to use those to validate the authenticity and integrity of the software

before installing or executing it. Code signing was conventional technique by November 1994 (¶¶ A98-A100, B74), using it was a standard practice, especially for software hosting financial transactions or providing a critical service. *See generally* Ex. 1057 (Dyad) at 132-35. Kahn explains that it is advantageous to use this technique to validate the authenticity and integrity of content distributed using its system, and it would have been obvious to extend this well-known technique to the software operating the server systems as well.

D143. Kahn shows that software programs can be distributed to end-users via its repository servers. *See* ¶ D32, *above*. Kahn shows that, when a repository provides a user with digital content, the content is signed by the repository. *Id.* at 26:36, 28:18. To validate the digital signature, the end-user uses the repository's certificate, which either is transmitted with the digital work or was previously provided to the end-user. *See id.* at 25:26-28 (requesting system can validate repository's signature and therefore it must have the repository's digital certificate); *id.* at 24:54-62 ("The requesting system signs and sends the following . . . message 772 to the repository: . . . [T]he requesting system or user's public key certificate chain, which will be used to verify the requestor's identity. Other less efficient methods can also be used, e.g. the repository could be given sufficient information (the distinguished name) about the user to obtain the certificate chain from another database. This is needed in the event the repository

needs to bill for providing the Terms and Conditions.”). Logically, an end-user would then validate the software’s digital signature before installing it.

D144. Thus, Kahn shows the repository possesses behavioral integrity.

b. The Linn End-User Devices Are “Repositories”.

D145. In the Linn system, content in an information object can be rendered only by the rendering software running on a client’s device. Ex. 1058 (Linn) at 13, 14, 15-16.

D146. Linn explains that the content in the information object is hidden using cryptographic techniques, and it is “difficult, if not impossible” for a user to extract the content from the object without the rendering software. *Id.* at 13; *id.* at 14, 15-16.

D147. The Linn rendering software, along with the techniques used to create the sealed envelopes, prevents unauthorized users from accessing the content. Ex. 1001 (859) at 12:16-18 (“repositories prevent access to the raw data by general devices and can test explicit rights and conditions before copying or otherwise granting access”), 13:10-12 (“A repository has both a hardware and a functional embodiment.”).

D148. Devices that run the Linn rendering software therefore possesses physical integrity. *See, e.g.*, Ex. 1001 (859) at 11:67-12:5 (physical integrity mean the repository “never allows non-trusted systems to access the digital works

directly.”), 12:16-18 (“repositories prevent access to the raw data by general devices and can test explicit rights and conditions before copying or otherwise granting access”).

D149.Linn explains that the only way content can be obtained and exchanged is by using the rendering software, but does not describe a protocol for exchange. *See* ¶¶ D70, D74-D77, *above*; Ex. 1058 (Linn) at 15.

D150. As I explained above, a person of ordinary skill would have considered it obvious to integrate the Kahn repository servers and the Linn client devices running the Linn rendering software by transmitting data between those devices using the secure transaction protocols described in Kahn; doing so would cause ensure both categories of devices would possess communications integrity within the integrated scheme. Ex. 1018 (Kahn) at 4:25-36, 3:53-4:16, 10:29-38, 10:44-48, 24:34-36, 23:48-60.

D151. Consequently, the Linn end-user devices running its rendering software would use the exchange of digital certificates, digital signatures, encryption, and nonces (random-value tags) to communicate with the Kahn repository servers in the combined Kahn/Linn system. *See* ¶¶ D21-D30, D33-D48, *above*.

D152. Therefore, both the repository servers and client devices running the Linn rendering software in a combined Kahn/Linn system would have

communications integrity. *See* ¶¶ D105-D118, *above*; *see, e.g.*, Ex. 1001 (859) at 12:24-33 (repositories can use “security measures involving encryption, exchange of digital certificates, and nonces”).

D153.Linn explains that rendering software is “widely available; e.g., distributed by publishers, information service providers, computer manufacturers.” Ex. 1058 (Linn) at 14. In the combined Kahn/Linn system, the rendering software logically would be provided by the entity operating the Kahn repository. Likewise, it would be logical that the rendering software would be digitally signed and distributed with a digital certificate. For example, the repository server’s certificate could be included with the software or it could be obtained separately by the end user. *See* Ex. 1018 (Kahn) at 24:54-62 (“The requesting system signs and sends the following . . . message 772 to the repository: . . . [T]he requesting system or user's public key certificate chain, which will be used to verify the requestor's identity. Other less efficient methods can also be used, e.g. the repository could be given sufficient information (the distinguished name) about the user to obtain the certificate chain from another database. This is needed in the event the repository needs to bill for providing the Terms and Conditions.”).

D154. Both Kahn and Linn show distribution of content with digital signatures and certificates to validate the authenticity and integrity of digital content. Ex. 1018 (Kahn) at 10:18-38, 25:26-28, 26:36, 28:18, 1:24-33, 6:65-7:12,

13:40-46; Ex. 1058 (Linn) at 15-16. Given the central role the rendering software plays in enforcing usage rights, it would have been obvious to distribute the rendering software with a certificate from the repository server to enable the recipient to authenticate its source and integrity.

D155. Kahn in view of Linn therefore would have made it obvious to configure the rendering software running on client devices to possess behavioral integrity.

3. A Kahn / Linn System Would Use “Authorization Objects”

D156. An “authorization object” is “a digital work in a file of a standard format” that is used “to receive the digital content and to use the digital content.”

D157. In the combined Kahn/Linn system suggested to the person of ordinary skill, the digitally signed authorization message from the Kahn payment server is an “authorization object” within the meaning of the claims. Ex. 1018 at 23:23-45.

D158. Kahn shows that each authorization message is digitally signed by the payment server and contains a standard set of data (*i.e.*, the message is a “*file of a standard format*”), such as an “Account Number, Credit Limit Amount, Time Limit, Categories of Use, List of authorized users . . . , and [t]he requesting system or user’s public key certificate chain, which will be used to verify the requestor's identity. . . .” Ex. 1018 at 23:24-45.

D159. Each authorization message also includes “usage rights” specifying how it can be used. *See ¶¶ D26-D28, above.* Each includes a “manner of use”: the “categories of use” field specifies whether the authorization message can be used to purchase different types of content (*e.g.*, used to purchase video or books). Each includes “conditions”: the “Credit Limit Amount” and “Time Limit” fields limit the types of purchases that can be made. Ex.1018 (Kahn) at 23:24-38. Thus, each authorization message described in Kahn is an “a digital work in a file of a standard format.”

D160. The Kahn authorization message is required “to receive [] digital content,” because without it, a user cannot purchase content from a repository server. Ex. 1018 at 23:1-25. In the combined Kahn/Linn system, an end-user would be required to have an authorization message “to use [] digital content.” Linn explains that when the rendering software renders content, it first checks a user ID field on the envelope to verify that it matches the ID of the end user (Ex. 1058 at 16), and a person of ordinary skill would have found it obvious to use the data in the digitally signed authorization message (*e.g.*, account number). Therefore, if the rendering software does not have the digitally signed authorization message for the user who purchased the content, it will not render the content because the user ID verification will fail.

D161. Thus, each authorization message is required both to receive and to use digital content, and each is an “authorization object” within the meaning of the claims.

F. A Kahn / Linn System Could Be Configured to Have Additional Security Capabilities Based on the Guidance in Dyad or Shear

D162. A person of ordinary skill would have considered implementing the combined Kahn/Linn system using devices with greater levels of security capabilities to have been obvious in view of Dyad or Shear, each of which describes hardware components that can improve the security of a system. It was well-known that incorporating tamper-resistant hardware into computer systems could increase the security of those systems. *See* ¶¶ A62-A90, *above*.

1. Dyad

D163. Both of Dyad and Linn were presented at the same conference and published in the Proceedings of that conference. *See* ¶¶ A115-A119, A123-A133, *above*. At that conference, Kahn also presented an overview of the system described in his patent. *See* ¶¶ A115-A119, A123-A133, *above*. A person of ordinary skill considering the Kahn and Linn references would have considered the complementary teachings of Dyad.

D164. Dyad explains that its secure coprocessor can be cost-effectively integrated into any type of computer (*e.g.*, a standard PC), including the end-user computers used in the Linn systems. Ex. 1057 (Dyad) at 121. A person of

ordinary skill in the art would have considered it to be obvious that adding a secure cryptographic processor to a system would improve security. The Dyad processor could be integrated into the Linn end-user devices, and it would have been obvious to configure the rendering software to leverage those security features.

D165. The Linn rendering software and end-user computer could readily be adapted to use the secure coprocessor to protect decryption keys and content extracted from an object. As Dyad explains, decryption keys and other data can be stored in the secure coprocessor, while encrypted content can be stored in memory.

D166. A person of ordinary also would have understood that the Dyad coprocessors could be incorporated into the Kahn repository servers, which would have improved the security of those systems. As Dyad explains, the coprocessor could enable verification of the server software to ensure it has not been modified, provide for the integrity and authenticity of transaction logs, provide secure payment processing, and provide encryption services. Ex. 1057 at 127-31, 139-41.

2. Shear

D167. A person of ordinary skill in the art would also have considered Shear to provide an example of possible design choices and implementation details that could be used to implement the combined Kahn/Linn system.

D168. Shear, Linn, and Kahn can all be used to create digital libraries. *See* Shear at 14:56-57 (“Personal Library software can be used to great advantage with

the present invention.”), 7:34-37 (“The present invention allows very large numbers of customers to acquire library disks at very low initial costs”), 6:38-62. A person of ordinary skill in the art would recognize these three systems to be in the same sub-field and would consider them together.

D169. Like Linn, Shear focuses on the end-user device side of a process for distributing and controlling an end-user’s usage of digital content.

D170. A person of ordinary skill in the art would understand that incorporating a tamper-proof “decrypting device and associated controller” into an end-user’s computer would improve the security of the Kahn/Linn scheme. Ex. 1055 (Shear) at Abstract, 3:49-59, 5:11-32, 6:38-47, 9:13-3.

D171. A person of ordinary skill would have found it logical and natural to provide users with rendering software and/or dedicated devices that perform the user-side of the Kahn protocols. Incorporating Linn rendering software and the Shear tamper-proof device into the same computer system would increase the security of the end-user’s system, and provide authors and content providers with greater assurance their content would be protected.

D172. In 1994, it was well-known that implementing security protocols in tamper-proof devices would increase the security of a computer system.

D173. In the combined system, the Linn rendering software could be configured to render the content and the security features could be integrated into

the Shear hardware device. Combining these systems would be a routine engineering task for a person of ordinary skill in the art, requiring only minor, logical modifications to each system to integrate them.

D174. Each of the many specific reasons for this conclusion discussed above with respect to Linn also apply to Shear.

D175. A person of ordinary skill in the art would understand that, for example, the Kahn system could be used to initially authorize the Shear device and to transmit digital content to it. The Shear device would be used for to meter and bill for content usage, by interfacing with the central system, as Shear describes. Moreover, a person of ordinary skill in the art would have considered it obvious that the “terms and conditions” that Kahn provides along with the digital content could be enforced by the Shear device in the same manner that the Shear device enforces the other restrictions described by Shear, including whether displaying and/or printing are allowed, time-based limits, quantity-based limits, user-defined limits, and so forth.

D176. In addition, a person of ordinary skill in the art would understand that the security mechanisms of Kahn and Shear are compatible. Kahn and Shear each describe use of highly secure and highly configurable systems. It would have required only routine effort using known techniques for a person of ordinary skill in the art to modify Shear to include additional security features, such as use of a

digital certificate to communicate with the centralized system of Kahn over a secure (encrypted) communications channel.

D177. Shear's system is flexible. For example, Shear describes the ability to perform functions using software, hardware, or a combination of the two, the ability to interface with remote central facilities, the ability to use formulate different database access requests, and the ability to use different data transport protocols and encryption schemes. Ex. 1055 (Shear) at 20:33-21:64, 19:27-54, 11:11-23, 14:43-45, 16:42-44. This flexibility would have made it even more obvious to a person of ordinary skill in the art that Shear's end-user device could be easily integrated with the Kahn centralized management system.

G. Kahn in view of Linn Would Render the Stefik Patents Obvious to a Person of Ordinary Skill

1. The Independent Claims of the '859 Patent

D178. The independent claims the '859 patent describe a “rendering system” that can access content over a network and render content in accordance with usage rights attached to that content. A “rendering device” is coupled to a “distributed repository,” and the distributed repository can request content from another repository and it enforces usage rights when content is rendered by the rendering device.

D179. Kahn describes a “distributed system” for distributing digital objects (“content”) to end-users. Ex. 1018 (Kahn) at 1:11-21, 4:46, 5:61-6:5, 6:25-45,

6:55-58, 7:28-36, 7:60-64, 8:20-30, 8:66-9:19, 22:28-34, 22:59-60, 24:34-25:3, 25:14-26:22, 26:29-38; Figs. 1, 23-27. Linn describes “rendering software” that is installed on a user’s computer and can be used to render the content (“rendering system”).Ex. 1058 (Linn) at 11, 13-16, 20. When a user requests to access content, the rendering software will check the usage information on the envelope and will permit the user to access the content only in accordance with usage information (“render[s] content in accordance with usage rights”).

D180. The rendering software installed on an end-user’s computer is a “distributed repository” that has both a “requester” “server mode of operation.” The rendering software can request to purchase and retrieve content from the Kahn repositories (“requester mode of operation”). The rendering software also receives requests to access or render content from a user, and will permit a user to access the content in accordance with the terms and conditions specified by the author (“server mode of operation”). The Kahn repository is another distributed repository because it is a distributed server system that uses digital certificates to communicate over a network.

D181. If a particular repository server does not have a copy of the digital content requested by a user, the repository will return an error message. Ex. 1018 at 25:10-13. Instead of returning an error message, it would have been an obvious

variation to configure the repository server to request the content from another server, as this was a common feature of distributed systems.

D182. Thus, it would have been obvious to configure the Kahn repository servers to have a “*requester mode*” of operation and to be a “*distributed repository*.”

D183. The processes I described above disclose all the limitations of claim 1 of the '859 patent. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis. I describe the combination of Kahn and Linn, which is applicable to all claims, in paragraphs *See* ¶¶ D95-D118.

<u>'859 Patent: Claim 1</u>	
1. A rendering system adapted for use in a distributed system for managing use of content, said rendering system being operative to rendering content in accordance with usage rights associated with the content, said rendering system comprising:	<p><u>“distributed system for managing use of content”</u></p> <p><u>Kahn</u> describes a distributed server system. ¶¶ D1-D4, D14-D21</p> <p><u>“rendering system”</u></p> <p>Computer running the <u>Linn</u> rendering software. ¶¶ D33, D47-D48, D50-D51, D55-D56, D70-D77</p> <p><u>“rendering content in accordance with usage rights”</u></p> <p>The rendering software enforces usage rights on the envelope. ¶¶ D5-D13, D58-D62, D70-D77, D111-D114, D119-D129</p>
[1.a] a rendering device configured to	<u>“rendering device”</u>

render the content; and	Computer running the <u>Linn</u> rendering software. ¶¶ D50-D51, D55-D56, D70-D77
[1.b] a distributed repository coupled to said rendering device and including a requester mode of operation and server mode of operation,	<p><u>“distributed repository”</u></p> <p>The <u>Kahn</u> distributed server system. ¶¶ D1-D4, D14-D21, D33-D48, D131-D144, D154-D155</p> <p>In the combined system, the rendering software is a “distributed repository” that can interact with the server system. ¶¶ D145-D155</p> <p><u>“mode[] of operation”</u></p> <p><i>See [1.c] & [1.d] below</i></p>
[1.c] wherein the server mode of operation is operative to enforce usage rights associated with the content and permit the rendering device to render the content in accordance with a manner of use specified by the usage rights,	<i>See [1.] above (“rendering content in accordance with usage rights”)</i>
[1.d] the requester mode of operation is operative to request access to content from another distributed repository, and	<i>See [1.] above (“rendering content in accordance with usage rights”)</i>
[1.e] said distributed repository is operative to receive a request to render the content and permit the content to be rendered only if a manner of use specified in the request corresponds to a manner of use specified in the usage rights.	<i>See [1.] above (“rendering content in accordance with usage rights”)</i>

D184. Independent claims 29 and 58 of the '859 patent are identical to claim 1, except claim 29 specifies a method and claim 58 specifies software. Kahn and
Petitioner Apple Inc. - Ex. 1013, p. 296

Linn disclose systems, methods, and software for implementing their systems. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'859 Patent: Claim 29</u>	<u>'859 Patent: Claim 58</u>	
29. A rendering method adapted for use in a distributed system for managing use of content, and operative to render content in accordance with usage rights associated with the content, said method comprising:	58. A computer readable medium including one or more computer readable instructions embedded therein for use in a distributed system for managing use of content, and operative to render content in accordance with usage rights associated with the content, said computer readable instructions configured to cause one or more computer processors to perform the steps of:	See Claim 1. <u>“computer readable instructions”</u> Rendering software ¶¶ D70-D77
[29.a] configuring a rendering device to render the content;	[58.a] configuring a rendering device to render the content;	See Claim 1.
[29.b] configuring a distributed repository coupled to said rendering device to include a requester mode of operation and server mode of operation;	[58.b] configuring a distributed repository coupled to said rendering device to include a requester mode of operation and server mode of operation;	See Claim 1.
[29.c] enforcing usage rights associated with the content and permitting the rendering device to render the content in accordance with a manner	[58.c] enforcing usage rights associated with the content and permitting the rendering device to render the content in accordance with a manner	See Claim 1.

of use specified by the usage rights, when in the server mode of operation;	of use specified by the usage rights, when in the server mode of operation;	
[29.d] requesting access to content from another distributed repository, when in the requester mode of operation; and	[58.d] requesting access to content from another distributed repository, when in the requester mode of operation; and	<i>See Claim 1.</i>
[29.e] receiving by said distributed repository a request to render the content and permitting the content to be rendered only if a manner of use specified in the request corresponds to a manner of use specified in the usage rights.	[58.e] receiving by said distributed repository a request to render the content and permitting the content to be rendered only if a manner of use specified in the request corresponds to a manner of use specified in the usage rights.	<i>See Claim 1.</i>

D185. It is my opinion that Kahn in view of Linn renders claims 1, 29, and 58 of the '859 patent obvious.

D186. If the claims of the Stefik patents are interpreted to require particular hardware components such as a tamper-resistant module or secure coprocessor, the claims of the '859 patent still are obvious in view of Kahn, Linn, and Dyad. *See ¶¶ D163-D166, above.*

2. Analysis of the Dependent Claims of the '859 Patent

a. Claims 3, 31, 60 and Claims 5, 33, 62 of the '859 Patent

D187. Claims 3, 31, and 60 depend from claims 1, 29, and 58, respectively, and specify that “said repository comprises means for storing the content.”

D188. Claims 5, 33, and 62 depend from claims 3, 31, and 60, respectively, and specify that the “means for storing” comprises “means for storing content after rendering.”

D189. Linn explains that content is stored on the end-user’s computer. **“Information should be stored and exchanged in standardized but device-independent forms.”** Ex. 1058 (Linn) at 13-14; *id.* at 13 (the required protection mechanisms are implemented “in **computer systems** as part of the application programs which deliver services to users”). A person of ordinary skill would understand that the rendering software (*i.e.*, “repository”) is part of a computer systems that has memory and allows content to be stored. Linn shows content is stored both before and after rendering.

D190. It is my opinion that Kahn in view of Linn renders claims 3, 31, and 60 and claims 5, 33, and 62 of the ’859 patent obvious.

b. Claims 8, 36, 65 and Claims 13, 40, 69 of the ’859 Patent

D191. Claims 8, 36, and 65 depend from claims 5, 33, and 62, respectively, and specify that the content to be rendered comprises “video.”

D192. Claims 13, 40, and 69 depend from claims 1, 29, and 58, respectively, and specify that the “rendering device” comprises a “video system.”

D193. Linn shows that rendering software is used to distribute and render video content. Rendering video content is a built-in feature of the Linn rendering

software. Ex. 1058 at 13-14, 15 (“**rendering of the information on a video display or printer** [is a] built-in function[] of the rendering software . . .”), 20.

D194. A person of ordinary skill would understand that a computer that can play video content is a “video system.”

D195. The Kahn repository servers also can be used to distribute video content. Ex. 1018 (Kahn) at 22:59-60 (digital object can be video), 1:17-21, 4:46 (“Any kind of digital object may be dealt with.”).

D196. It is my opinion that Kahn in view of Linn renders claims 8, 36, and 65 and claims 13, 40, and 69 of the ’859 patent obvious.

c. Claims 15, 42, 71 of the ’859 Patent

D197. Claims 15, 42, and 71 depend from claims 1, 29, and 58, respectively, and specify that the “rendering device” comprises a “computer system” and the “repository” comprises “software executed on the computer system.”

D198. The Linn rendering software runs on an end-user’s computer and the rendering software (“*software executed on the computer system*”) enforces usage rights and provides physical, communications, and behavioral integrity.

D199. It is my opinion that Kahn in view of Linn renders claims 15, 42, and 71 of the ’859 patent obvious.

d. Claims 16, 43, 72 of the '859 Patent

D200. Claims 16, 43, and 72 depend from claims 1, 29, and 58, respectively, and specify that an “execution engine” is coupled to the repository and the repository will “permit said execution device to execute a computer program only in a manner specified by the usage rights.”

D201. The Kahn repository servers can be used to distribute software and video games. Ex. 1018 (Kahn) at 1:11-13, 1:17-25 (“a digital object could include programs and data which, though not directly a representation of the text of a work, enable the delivery over a network and the subsequent reproduction on a computer screen of selected portions of the text of the work.”), 4:46, 5:66-6:5 (digital objects include “video games”).

D202. The Linn rendering software can process any “machine-processable form of digital information,” (Ex. 1058 at 20), and this would include programs that enable content to be rendered and video games. A person of ordinary skill in the art would understand that video games are a type of software.

D203. Kahn in view of Linn shows that the end-user’s computer (“*execution engine*”) running the rendering software can be used to execute software such as programs that enable content to be rendered and video games (“*execute a computer program*”).

D204. It is my opinion that Kahn in view of Linn renders claims 16, 43, and 72 of the '859 patent obvious.

e. **Claims 19, 46, 75 and Claims 20, 47, 76 of the '859 Patent**

D205. Claims 19, 46, and 75 depend from claims 1, 29, and 58, respectively, and specify that the “manner of use is a manner of displaying.”

D206. Claims 20, 47, and 76 depend from claims 1, 29, and 58, respectively, and specify that the “manner of use is a manner of playing.”

D207. Displaying a work is a manner of “playing” within the context of the claims includes processes such as “playing digital movies, playing digital music, playing a video game, running a computer program, or displaying a document on a display.” Ex. 1001 (859) at 18:49-53.

D208. Linn explains an author can grant the right to allow a work to be displayed or played on a monitor, or to display a digital work by printing it: “Processing software employed by a user should *display* or print the materials in *an appropriate form* given the constraints of the user’s *video display* and/or *printer*.” Ex. 1058 (Linn) at 13-14, 15 (“*The number of printed copies allowed* is a function of the fee paid.. . . Materials may be displayed on a video display an unlimited number of times by the user identified on the envelope.”) (emphases added).

D209. Whether a digital work can be displayed on a monitor or printed is both a manner of displaying the work and a manner of playing the work.

D210. It is my opinion that Kahn in view of Linn renders claims 19, 46, and 75 and claims 20, 47, and 76 of the '859 patent obvious.

f. Claims 21, 48, 77 of the '859 Patent Would Have Been Obvious Considering Dyad

D211. Claims 21, 48, and 77 depend from claims 1, 29, and 58, respectively, and specify that “the rendering device and the repository are integrated into a secure system having a secure boundary.”

D212. As I explained above, it would have been obvious to incorporate the secure coprocessor described in Dyad into the combined Kahn/Linn systems.

D213. By incorporating the secure coprocessor into the end-user's computer system, the user's computer and the rendering software are part of a secure system that has a secure boundary.

D214. For the same reasons, it is my opinion that Kahn in view of Linn and Dyad renders obvious the other independent and dependent patent claims that I address in this report, if such claims are interpreted to require particular hardware components such as a tamper-resistant module or secure coprocessor.

g. Claims 24, 51, 81 of the '859 Patent

D215. Claims 24, 51, and 81 depend from claims 1, 29, and 58, respectively, and specify that the rendering system “comprises means for” or is “configured to”

“communicat[ing] with a master repository for obtaining an identification certificate for the repository.”

D216. A person of ordinary skill in the art would have configured the Linn rendering software to implement the user registration process described in Kahn.

D217. Kahn explains that “[b]efore a user can retrieve any objects for which payment is required, the user must first establish an account with a payment server system.” Ex. 1018 at 22:28-30, 22:45-67 (“The setup-new-account message . . . is sent [] to the payment server . . .”).

D218. To establish an account, a user must first obtain a digital certificate (“*identification certificate*”) from a certificate authority (“*master repository*”), and then obtain a digitally signed account message from the payment server (also a “*master repository*”). *See id.* at Fig. 4.

D219. The Linn rendering software is installed on a computer and is configured to communicate over a network. The computer thus contains a network card that allows it to communicate with other computers (“*means for . . . communicating with a master repository*”). Ex. 1058 at 13, 15.

D220. It is my opinion that Kahn in view of Linn renders claims 24, 51, and 81 of the ’859 patent obvious.

3. The Claims of the '072 Patent

D221. The independent claims the '072 patent (claims 1, 10, and 18) describe similar processes. Each independent claim specifies a “document platform” that requests and/or receives content and “at least one usage right” from a “document repository.” The document platform stores the content and the usage right in separate files, and it renders the content but only in accordance with the usage right (or rights). Claims 10 and 18 each additionally specify that the document repository authenticates the document platform. Claim 10 additionally specifies that the document repository stores the content and the at least one usage right in separate files.

D222. The combination of Kahn and Linn disclose the processes described in the '072 independent claims. Kahn and Linn describe techniques for distributing digital content (“digital document”) and enforcing compliance with usage information attached to that content by an author or distributor. Kahn discloses a repository (“document repository”), and the Linn rendering software installed on an end-user’s computer (“document platform”) is used to acquire and use content. In the combined system, content is distributed in a sealed envelope that contains usage rights information (“at least one usage right”).

D223. In the combined system, the rendering software implements the Kahn transaction protocols, which allows an end-user to request, purchase, and receive

content from a repository (“retrieving . . . a digital document and at least one usage right”). The Kahn repository will receive a request from a user (“receiving a request”), will authenticate the user’s digital certificate and signed account information, and if the purchase transaction is successful, it will transfer content in a sealed envelope to the customer (“transferring the digital document”). Linn shows that the envelope and digital content are separate files, and that the envelope is attached to the content using a digital signature. After receiving content from a repository, the rendering software will store the content and envelope in memory on an end-user’s computer (“storing . . . in separate files”).

a. Claim 1 of the '072 Patent

D224. The processes I described above disclose all the limitation of claim 1 of the '072 patent. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'072 Patent: Claim 1</u>	<u>Relevant Concepts and Paragraphs</u>
1. A method for securely rendering digital documents, comprising:	<u>“digital document”</u> The <u>Kahn</u> compound objects could be integrated into a single envelope using a <u>Linn</u> SGML content envelope. ¶¶ D114-D115.
[1.a] retrieving, by a document platform, a digital document and at least one usage right associated with the digital document from a document repository, the at least one usage right specifying a manner of use indicating	<u>“retrieving . . . a digital document”</u> The rendering software allows a user to request and receive digital content from a repository. ¶¶ D95-D118

<p>the manner in which the digital document can be rendered;</p>	<p><u>“document repository”</u></p> <p>The <u>Kahn</u> repository server. ¶¶ D1-D4, D14-D21, D33-D48, D131-D144, D154-D155</p> <p><u>“the at least one usage right specifying a manner of use indicating the manner in which the digital document can be rendered”</u></p> <p>The repository server stores terms and conditions for each object. ¶¶ D5-D13.</p> <p>Content is transmitted to the user in the <u>Linn</u> envelope, and it would have been obvious to add fields to envelope for the terms and conditions (<i>e.g.</i>, perform, use, modify, transmit) described in <u>Kahn</u>. ¶¶ D58-D62, D111-D114, D119-D129</p>
<p>[1.b] storing the digital document and the at least one usage right in separate files in the document platform;</p>	<p>The rendering software stores content on an end-user’s computer. ¶¶ D70-D77.</p> <p><u>Linn</u> shows the digital content and the envelope are separate files. ¶¶ D64-D69</p> <p>In the combined system, the rendering software is a “repository” that can interact with the repository server. ¶¶ D145-D155.</p>
<p>[1.c] determining, by the document platform, whether the digital document may be rendered based on the at least one usage right; and</p>	<p>The rendering software will check the usage information on the envelope before permitting a user to access the content. ¶¶ D70-D77, D113-D129.</p>

[1.d] if the at least one usage right allows the digital document to be rendered on the document platform, rendering the digital document by the document platform.	The rendering software will check the usage information on the envelope before permitting a user to access the content. ¶¶ D70-D77, D113-D129.
---	--

b. Claim 10 of the '072 Patent

D225. The processes I described above disclose all the limitation of claim 10 of the '072 patent. I note that Claim 10.b specifies a document repository receives a “request for access” to content and in response it “transfers” content to the document platform. Claims 1 and 18 each specify a “request to transfer.” A person of ordinary skill would understand that a “request for access” to content would be satisfied by a “request to transfer” content. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'072 Patent: Claim 10</u>	<u>Relevant Concepts and Paragraphs</u>
10. A method for securely rendering digital documents, comprising:	<i>See</i> Claim 1.
[10.a] storing a digital document and at least one usage right in separate files in a document repository, wherein the at least one usage right is associated with the digital document;	The <u>Kahn</u> repository server. ¶¶ D1-D4, D14-D21, D33-D48, D131-D144, D154-D155. <u>Kahn</u> shows that a digital object contains an external properties record, which contains the “stated terms and conditions for access and usage of the object.” ¶¶ D5-D13.
[10.b] receiving a request from a document platform for access to the digital document;	The rendering software allows a user to request and receive digital content from a repository. ¶¶ D70-D77,

	<p>The repository will receive requests from users to access content. ¶¶ D33-D49.</p> <p>In the combined system, the rendering software is a “repository” that can interact with the repository server. ¶¶ D145-D155</p>
<p>[10.c] determining, by the document platform, whether the request may be granted based on the at least one usage right, the determining step including authenticating the document platform and determining whether the at least one usage right includes a manner of use that allows transfer of the digital document to the document platform;</p>	<p><u>“determining . . .”</u></p> <p>The Kahn repository server determines whether the request can be granted. <i>See</i> Claim 1.c.</p> <p>When the rendering software receives object, it will not accept the object if the usage information does not permit it. ¶ D56; <i>see also</i> ¶¶ D145-D155</p> <p><u>“authenticating the document platform”</u></p> <p>The <u>Kahn</u> repository will authenticate a user by validating the user’s digital certificate. ¶¶ D22-D26, D39-D49</p> <p><u>“a manner of use that allows transfer”</u></p> <p><u>Linn</u> shows the envelope contains fields that can permit redistribution. Content is transmitted to the user in the <u>Linn</u> envelope, and it would have been obvious to add fields to envelope for the terms and conditions (<i>e.g.</i>, perform, use, modify, transmit) described in <u>Kahn</u>. ¶¶ D58-D62, D111-D114, D119-D129</p>

[10.d] if the at least one usage right allows the transfer of the digital document to the document platform, transferring the digital document and the at least one usage right associated with the digital document to the document platform;	The <u>Kahn</u> repository will only provide a user with content in accordance with the terms and conditions specified by the author. ¶¶ D5-D9.
[10.e] storing the digital document and the at least one usage right in the document platform, wherein the at least one usage right is stored in a separate file from the digital document; and	See Claim 1.b.
[10.f] rendering the digital document by the document platform.	See Claim 1.d.

c. Claim 18 of the '072 Patent

D226. The processes I described above disclose all the limitation of claim 18 of the '072 patent. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'072 Patent: Claim 18</u>	<u>Relevant Concepts and Paragraphs</u>
18. A method for securely rendering digital documents, comprising:	See Claim 1.
[18.a.] receiving a request from a document platform to transfer a digital document from a document repository to the document platform, the digital document having at least one usage right associated therewith;	See Claim 10.b.
[18.b.] authenticating the document platform by accessing at least one identifier associated with the document platform or with a user of the document platform and determining whether the identifier is associated with at least one	See Claim 1.c & 10.c. <u>“authenticating . . . by accessing at least one identifier”</u> The <u>Kahn</u> repository will authenticate a

of the document platform and a user authorized to use the digital document;	<p>user by validating the user's digital certificate. ¶¶ D22-D27, D33-D45.</p> <p><u>“determining whether the identifier . . . is associated with . . . [an] authorized [document platform or user]”</u></p> <p>The <u>Kahn</u> repository will ensure a user is authorized by validating the user's signed account information (from the payment server). ¶¶ D22-D27, D33-D49.</p>
[18.c.] if the authenticating step is successful, determining whether the digital document may be transferred and stored on the document platform based on a manner of use included in the at least one usage right;	The <u>Kahn</u> repository will only provide a user with content in accordance with the terms and conditions specified by the author. ¶¶ D5-D9.
[18.d.] if the at least one usage right allows the digital document to be transferred to the document platform, transferring the digital document and the at least one usage right associated with the digital document to the document platform;	See Claim 10.d.
[18.e.] storing the digital document and the at least one usage right in the document platform, wherein the at least one usage right is stored in a separate file from the digital document;	See Claim 10.e.
[18.f.] determining, by the document platform, whether the digital document may be rendered based on the at least one usage right; and	See Claim 1.c.
[18.g.] if the at least one usage right allows the digital document to be	See Claim 1.d.

rendered on the document platform, rendering the digital document by the document platform	
--	--

D227. It is my opinion that Kahn in view of Linn renders claims 1, 10, and 18 of the '072 patent obvious.

D228. If the claims of the Stefik patents are interpreted to require particular hardware components such as a tamper-resistant module or secure coprocessor, the claims of the '072 patent still are obvious in view of Kahn, Linn, and Dyad. See ¶¶ D163-D166, *above*.

d. Claims 8, 16, and 24 of the '072 Patent

D229. Claims 8, 16, and 24 depend from claims 1, 10, and 18, respectively, and specify that “at least one part of the digital document and the at least one usage right are stored on a same device.”

D230. Linn shows that the rendering software stores content (“digital document”) and the envelope (“usage right”) on the end-user’s computer (“same device”). Ex. 1058 (Linn) at 15.

D231. It is my opinion that Kahn in view of Linn renders claims 8, 16, and 24 of the '072 patent obvious.

4. The Claims of the '956 and '007 Patents

D232. I am addressing the claims of the '956 and '007 patents together. Both the '956 and '007 claims are directed to the same basic processes for

transferring content between a sending device and recipient device. The '956 claims are directed toward client or recipient device, while the '007 claims are directed toward the server or sending device.

a. Claims 1, 7, and 13 ('956) and 1, 6, and 11 ('007)

D233. Kahn and Linn disclose the processes described in the '956 and '007 independent claims. Because the independent claims are highly similar, I have addressed them together below.

D234. Kahn describes a repository (“sending computing device”), which is a system for distributing digital objects (“content”) to end-users. Linn describes “rendering software” that is installed on a user’s computer and can be used to request and render content (“recipient computing device” or “apparatus”). When a user requests to access content, the rendering software will check the usage information on the envelope and will permit the user to access the content only in accordance with usage information specified on the envelope (“usage rights information”).

D235. In the tables below, I have mapped each claim element to the relevant paragraphs of my analysis. I describe the combination of Kahn and Linn, which is applicable to all claims, in paragraphs D95-D118.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[Claim 1. preamble] A computer-implemented	[Claim 1. preamble] A computer-implemented	<u>“recipient computing device” for “rendering</u>

method of rendering digital content by at least one recipient computing device in accordance with usage rights information, the method comprising:	method of distributing digital content to at least one recipient computing device to be rendered by the at least one recipient computing device in accordance with usage rights information, the method comprising:	<p><u>digital content . . . in accordance with usage rights”</u></p> <p>In the combined system, the rendering software is a “repository” that can interact with the repository server and renders content. ¶¶ D145-D155; <i>see also</i> ¶¶ D59-D62, D70-D77, D104, D109, D119-D128.</p> <p><u>“usage information”</u></p> <p>Content is transmitted to the user in the <u>Linn</u> envelope, and it would have been obvious to add fields to envelope for the terms and conditions (e.g., perform, use, modify, transmit) described in <u>Kahn</u>. ¶¶ D58-D62, D111-D114, D119-D129</p>
[Claim 7. preamble] A recipient apparatus for rendering digital content in accordance with usage rights information, the recipient apparatus comprising:	[Claim 6. preamble] A sending apparatus for distributing digital content to at least one recipient computing device to be rendered by the at least one recipient computing device in accordance with usage rights information, the sending apparatus comprising:	<i>See [Claim 1.preamble] above</i>

[Claim 13. preamble] At least one non-transitory computer-readable medium storing computer-readable instructions that, when executed by at least one recipient computing device, cause the at least one recipient computing device to:	[Claim 11. preamble] At least one non-transitory computer-readable medium storing computer-readable instructions that, when executed by at least one sending computing device, cause the at least one sending computing device to:	<i>See [Claim 1 preamble], above.</i>
--	--	---------------------------------------

D236. Kahn and Linn disclose systems, methods, and software for implementing their respective content distribution techniques. The Kahn repositories are a system of computers acting as server, and each computer contains a processor, memory, and software for enabling its functionality. Similarly, Linn shows rendering software that is installed on an end-user's computer, and the end-user's computer contains a processor, memory, and software (including the rendering software) for enabling its functionality. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[7.a.] one or more processors; and	[6.a.] one or more processors; and	<i>See [Claim 7.preamble] above</i>
[7.b.] one or more memories operatively coupled to at	[6.b.] one or more memories operatively	The computing devices in <u>Kahn</u> and

least one of the one or more processors and having instructions stored thereon that, when executed by at least one of the one or more processors, cause at least one of the one or more processors to:	coupled to at least one of the one or more processors and having instructions stored thereon that, when executed by at least one of the one or more processors, cause at least one of the one or more processors to:	<u>Linn</u> have processors and memories.
--	--	---

D237. Kahn shows that when a repository receives a request to access content, the request will contain the user’s digital certificate and the user’s account information, signed by a payment server. *See ¶¶ D33-D46, above.* The repository will validate the user’s digital certificate and the digitally signed account information. *See ¶¶ D27, D33-D46, above.* If the digital certificate and signed account information are valid, the repository determines that the user can be “trusted” (“determining if the . . . recipient computing device is trusted to receive the digital content”).

D238. If the purchase transaction is successful, the repository will transfer an content in an envelope containing the usage rights information to the rendering software (“receiving the digital content”). In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>’956 Patent</u>	<u>’007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[1.a.] receiving the digital content by the at	[1.a.] determining, by at least one sending	<u>“determination of trust”</u>

<p>least one recipient computing device from at least one sending computing device only if the at least one recipient computing device has been determined to be trusted to receive the digital content from the at least one sending computing device;</p>	<p>computing device, if the at least one recipient computing device is trusted to receive the digital content from the at least one sending computing device;</p>	<p>Repository server validates the client's digital certificate and digitally signed account information before sending content. ¶¶ D1-D2, D14-D21, D149-D161</p> <p><u>“sending computing device”</u></p> <p>The <u>Kahn</u> repository server system. ¶¶ D1-D4, D14-D21, D33-D48, D131-D144, D154-D155</p> <p><u>“recipient computing device”</u></p> <p>In the combined system, the rendering software is a “repository” that can interact with the repository server. ¶¶ D145-D155.</p>
<p>[7.c.] enable the receipt of the digital content by the recipient apparatus from at least one sending computing device only if the recipient apparatus has been determined to be trusted to receive the digital content from the at least one sending computing device;</p>	<p>[6.c.] determine if the at least one recipient computing device is trusted to receive the digital content from the sending apparatus;</p>	<p><i>See [Claim 1.a], above</i></p>
<p>[13.a.] receive the digital content from at least one sending computing device only if the at least</p>	<p>[11.a.] determine if the at least one recipient computing device is trusted to receive the</p>	<p><i>See [Claim 1.a], above</i></p>

one recipient computing device has been determined to be trusted to receive the digital content from the at least one sending computing device;	digital content from the at least one sending computing device;	
---	---	--

D239. As I explained above, Kahn will transfer the digital content (“sending the digital content”) and an envelope containing usage information (“sending the usage information”) only if the user can show it is trusted by presenting a digital certificate and the digitally signed account information. *See ¶ D27, above.* In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
N/A	<p>[1.b.] [11.b.] send[ing] the digital content, by the at least one sending computing device, to the at least one recipient computing device only if the at least one recipient computing device has been determined to be trusted to receive the digital content from the at least one sending computing device; and</p> <p>[6.d.] send the digital content, by the sending apparatus, to the at least one recipient computing device only if the at least one recipient computing device has been determined to be trusted to receive the digital content from the</p>	<i>See [Claim 1.a], above</i>

	sending apparatus; and	
	<p>[1.c.] [11.c.] send[ing] usage rights information indicating how the digital content may be rendered by the at least one recipient computing device, the usage rights information being enforceable by the at least on recipient computing device.</p> <p>[6.e.] send usage rights information indicating how the digital content may be rendered by the at least one recipient computing device, the usage rights information being enforceable by the at least on recipient computing device.</p>	<p><u>“usage rights . . . indicating how the digital content may be rendered”</u></p> <p>Content is transmitted to the user in the <u>Linn</u> envelope, and it would have been obvious to add fields to envelope for the terms and conditions (<i>e.g.</i>, perform, use, modify, transmit) described in <u>Kahn</u>. ¶¶ D58-D62, D111-D114, D119-D129</p>

D240. The rendering software is used to render content (“rendering the digital content”). For a user to render content, it necessarily must request a the software to render the it (“receiving a request . . . to render the digital content”).

D241. Before allowing a user access to content, the rendering software checks the fields on the envelope and enforces usage rights when a user attempts to render the content (“determining . . . whether the digital content may be rendered”). See ¶¶ D55-D57, D70-D77, *above*.

D242. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[1.b.], [7.d], [13.b] receiv[ing], [by the at least one recipient computing device,] a request to render the digital content;	N/A	See [Claim 1.preamble], <i>above</i>
[1.c.], [7.e], [13.c] determin[ing], based on the usage rights information, whether the digital content may be rendered by the at least one recipient computing device [apparatus]; and		See [Claim 1.preamble], <i>above</i>
[1.d.], [7.f], [13.d] render[ing] the digital content, [by the at least one recipient computing device,] only if it is determined that the content may be rendered by the at least one recipient computing device [apparatus].		See [Claim 1.preamble], <i>above</i>

D243. Therefore, it is my opinion that Kahn in view of Linn renders claims 1, 7, and 13 of the '956 patent obvious. And that Kahn in view of Linn renders claims 1, 6, and 11 of the '007 patent obvious.

D244. If the claims of the Stefik patents are interpreted to require particular hardware components such as a tamper-resistant module or secure coprocessor, the claims of the '956 and '007 patents still are obvious in view of Kahn, Linn, and Dyad. See ¶¶ D163-D166, *above*.

b. Claims 2, 8, 14 of the '956 Patent

D245. Claims 2, 8, and 14 of the '956 patent depend from claims 1, 7, and 13, respectively, and each specifies that the recipient device will “*deny[] the*

request and prevent[] rendering of the digital content . . . if it is determined that the digital content may not be rendered”

D246. As I explained above, before allowing a user access to content, the rendering software checks the fields on the envelope and enforces usage rights when a user attempts to render the content (“determining . . . whether the digital content may be rendered”). See ¶¶ D55-D57, D70-D77, *above*.

D247. Therefore, Kahn in view of Linn renders claims 2, 8, and 14 of the ’956 patent obvious.

c. **Claims 3, 9, 15 of the ’956 Patent and 2, 7, and 14 of the ’007 Patent**

D248. Claims 3, 8, and 15 of the ’956 depend from claims 1, 7, and 12, respectively, and each specifies the usage rights information includes a “*condition under which the content can be rendered.*”

D249. Claims 2, 7, and 14 of the ’007 depend from claims 1, 6, and 11, respectively, and each specifies the usage rights information includes a “*condition under which the content can be rendered.*”

D250. Kahn and Linn each show terms and conditions that include conditions (*e.g.*, a print counter, maximum number of copies), and a person of ordinary skill would retain those features in the combined system. See ¶¶ D119-D126, *above*; Ex. 1018 at 25:29-31; Ex. 1019 at 15-16.

D251. Therefore, it is my opinion that Kahn in view of Linn renders claims 3, 9, and 15 of the '956 patent obvious.

D252. Therefore, it is my opinion that Kahn in view of Linn renders claims 2, 7, and 14 of the '007 patent obvious.

d. Claims 4, 10, 16 ('956) and 3, 8, 16 ('007)

D253. Claims 4, 10, and 16 of the '956 depend from claims 1, 7, and 13, respectively, and each specifies that the step of “*receiving the digital content comprises*” two additional steps by the recipient device: requesting an “authorization object” and “receiving” it if authorized.

D254. Claims 3, 8, and 16 of the '007 depend from claims 1, 6, and 11, respectively, each and specifies that the step of making “*the determination of trust*” comprises two additional steps by the sending device: receiving a “request . . . for an authorization object” and transmitting the object “when it is determined that the request should be granted.”

D255. As I explained above (¶¶ D104-D108, D149-D152), the Linn rendering software is configured to implement the account registration and transaction protocols described in Kahn. In the combined system, before a user can request content from a repository, the user must establish an account with a payment server. The user will send a “setup-new-account message” (“requesting

an authorization object”) to the payment server that contains the user’s digital certificate and other data (“receiving a request . . . for an authorization object”).

D256. The payment server will validate all the information in the message. If the payment server determines that authorization should not be granted, it will send an error message to the user. Ex. 1018 (Kahn) at 23:1-20. If the payment server determines authorization should be granted (“determin[ing]” whether the request “should be granted”), it will return to the user a digitally signed “new-account-response message” containing account information (“receiving [transmitting] an authorization object”). *Id.* at 23:21-45.

D257. The signed account information message is required to be able to connect to a repository and receive content.

D258. Linn shows that each information envelope could have a user ID field on it, and that the rendering software will check the user’s ID against that field before allowing the user to access the content. Ex. 1058 (Linn) at 15-16. Linn does not, however, explain how users are assigned ID values. It would have been obvious to configure the rendering software to use an identifier in the digitally signed authorization message from the payment server as the user ID (*e.g.*, account number or an identifier from a digital certificate), and the validate the user ID on an envelope by checking it against the signed account information. Ex. 1018

(Kahn) at 23:24-38. Therefore, the authorization object is required to be able to use the content.

D259. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[4.a.], [10.a.], [16.a.] requesting an authorization object for the at least one recipient computing device [receiving apparatus] to make the digital content available for use, the authorization object being required to receive the digital content and to use the digital content; and	[3.a.], [8.a.], [16.a.] receiving a request from at least one recipient computing device for an authorization object required to render the digital content; and	<u>“authorization object”</u> Digitally signed authorization message from the payment server <i>See ¶¶ D33-D45, D156-D161</i>
[4.b.], [10.b.], [16.b.] receiving the authorization object if it is determined that the request for the authorization object should be granted.	[3.b.], [8.b.], [16.b.] transmitting the authorization object to the at least one recipient computing device when it is determined that the request should be granted.	<u>“authorization object”</u> Digitally signed authorization message from the payment server <i>See ¶¶ D33-D45, D156-D161</i>

D260. Therefore, it is my opinion that Kahn in view of Linn renders claims 4, 10, and 16 of the '956 patent obvious.

D261. It is my opinion that Kahn in view of Linn renders claims 3, 8, and 16 of the '007 patent obvious.

e. Claims 5, 11, 17 ('956) and Claims 4, 9, 14 ('007)

D262. Claims 5, 11, and 17 of the '956 patent depend from claims 1, 7, and 13, respectively, each specifies a method, apparatus, or software where the step of “receiving the digital content” or “enabling the receipt of the digital content” comprises three additional steps by the recipient computing device: generating a “registration message,” exchanging a session key with the sending device, and conducting a session where it receives content from the sending device.

D263. Claims 4, 9, and 14 of the '007 patent depend from claims 1, 6, and 13, respectively, each specifies a method, apparatus, or software where the step of making “the determination of trust” comprises four additional steps by the sending computing device: receiving a “registration message” from a recipient device, validating the authenticity of the recipient device, exchanging a session key with the recipient computing device, and conducting a session where it sends content to the recipient device.

D264. As with the independent claims, these '956 and '007 claims are directed to the same process, but the '956 claims cover the client device and the '007 claims cover the server device. These additional steps simply add standard cryptographic techniques that were well-known in the art.

D265. In a purchase transaction with a repository, the user obtains a handle for the content the user wants to purchase and then sends a request to initiate a session with a repository. *See* ¶¶ D38-D43, *above*. The repository will return a message to the requesting system that includes a “random-value tag”, which is a nonce that will be used to ensure the requesting system is authentic and not replaying messages. *See* ¶ D30, *above*.

D266. Next, the requesting system will request a copy of the terms and conditions for the content by sending a message (“registration message”) to the repository that contains the user’s digital signature over the random-value tag, the user’s digital certificate (“identification certificate”), the account information signed by the payment server, and a “unique tag” generated by the system (“registration identifier”). *Id.* at 24:47-65 (emphasis added). The message also includes a “unique key” that will be used as a session key (“exchanging messages including . . . [a] session key”). *Id.* at 25:20-21 (explaining that the repository returns to the user “[t]he user-assigned unique key, which was received in the request-terms-and-conditions message”); *id.* at 10:29-38. The unique key will be used to encrypt communications (“conducting a secure transaction using the session key”). *See* ¶¶ D33-D49, *above*; *see also id.* at 10:29-38.

D267. The repository will use the user's digital certificate to validate the user's digital signature on the message ("validating the [device's] authenticity"). If the signature is valid, the repository will initiate a session.

D268. The repository will send the user the terms and conditions. The user can select the terms the user wishes to purchase, and send the request to the repository. The will send the request to the payment server for authorization. If the transaction is authorized, the repository will send the user "the object, signed by the repository" and the "portions of the relevant terms and conditions" ("sending [receiving] the digital content"). *Id.* at 26:31-38.

D269. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[5.a.], [11.a.], [17.a.] generating a registration message, the registration message including an identification certificate of the recipient computing device [apparatus] and a random registration identifier, the identification certificate being certified by a master device;	[4.a.] [9.a.] [14.a.] receiving a registration message from the at least one recipient device, the registration message including an identification certificate of the recipient computing device and a random registration identifier, the identification certificate being certified by a master device;	<u>"registration message"</u> See ¶¶ D156-D161
N/A	[4.b.] [9.b.] [14.b.]	See ¶¶ D14-D48,

	validating the authenticity of the at least one recipient device;	D134-D140, D156-D161
[5.b.], [11.b.], [17.b.] exchanging messages including at least one session key with at least one provider computing device, the session key to be used in communications during a session; and	[4.c.] [9.c.] [14.c.] exchanging messages including at least one session key with the at least one recipient device, the session key to be used in communications; and	See ¶¶ D14-D48, D134-D140, D156-D161
[5.c.], [11.c.], [17.c.] conducting a secure transaction using the session key, wherein the secure transaction includes receiving the digital content.	[4.d.] [9.d.] [14.d.] conducting a secure transaction using the session key, wherein the secure transaction includes sending the digital content to the at least one recipient device.	See ¶¶ D14-D48, D134-D140, D156-D161

D270. It is my opinion that Kahn in view of Linn renders claims 5, 11, and 17 of the '956 patent obvious.

D271. It is my opinion that Kahn in view of Linn renders claims 4, 9, and 14 of the '007 patent obvious.

D272. If the claims of the Stefik patents are interpreted to require particular hardware components such as a tamper-resistant module or secure coprocessor, the dependent claims of the '956 and '007 patents still are obvious in view of Kahn, Linn, and Dyad. See ¶¶ D163-D166, *above*.

f. Claims 6, 12, 18 ('956) and Claims 5, 10, 15 ('007)

D273. Claims 6, 12, and 18 of the '956 patent depend from claims 5, 11, and 17, respectively, each specifies the method, apparatus, or software comprises two additional steps. The two additional steps in each of claims 6, 12, and 18 are identical. These steps simply add standard cryptographic techniques that were well-known in the art.

D274. Claims 5, 10, and 15 of the '007 patent depend from claims 1, 9, and 14, respectively, each specifies a method, apparatus, or software where the step of making “validating” comprises four additional steps. The four additional steps in each of claims 5, 10, and 15 are identical.

D275. As with the independent claims, these '956 and '007 claims are directed to the same process, but the '956 claims cover the client device and the '007 claims cover the server device. These additional steps simply add standard cryptographic techniques that were well-known in the art.

D276. In a purchase transaction with a repository, the user obtains a handle for the content the user wants to purchase and then sends a request to initiate a session with a repository. *See ¶¶ D38-D43, above.* The repository will return a message to the requesting system that includes a “random-value tag”, which is a nonce that will be used to ensure the requesting system is authentic and not

replaying messages (“sending [receiving] a message to test authenticity”) (“generating [receiving] a message . . . including a nonce”). *See* ¶ D30, *above*.

D277. Next, the requesting system will request a copy of the terms and conditions for the content by sending a message to the repository that contains the user’s digital signature over the random-value tag (“processing the generated message”), the user’s digital certificate (“identification certificate”), and the account information signed by the payment server. *Id.* at 24:47-65 (emphasis added). The message also includes a “unique key” that will be used as a session key (“exchanging messages including . . . [a] session key”). *Id.* at 25:20-21 (explaining that the repository returns to the user “[t]he user-assigned unique key, which was received in the request-terms-and-conditions message”); *id.* at 10:29-38. The unique key will be used to encrypt communications (“conducting a secure transaction using the session key”). *See* ¶¶ D22-D28, *above*; *see also id.* at 10:29-38.

D278. The repository will validate the user’s digital certificate (“verifying the identification certificate”), and then it will use the user’s digital certificate to validate the user’s digital signature on the message, which includes the random value-tag (“verifying” the message was “correctly processed”). If the signature is valid, the repository will initiate a session. As part of this process, the repository may send the user another random-value tag to verify the device’s identity

(“sending [receiving] a message to test authenticity”) (“generating [receiving] a message . . . including a nonce”).

D279. The repository will send the user the terms and conditions. The user can select the terms the user wishes to purchase, and send the request to the repository. The will send the request to the payment server for authorization. If the transaction is authorized, the repository will send the user “the object, signed by the repository” and the “portions of the relevant terms and conditions.” *Id.* at 26:31-38.

D280. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
N/A	[5.a.] [10.a.] [15.a.] verifying the identification certificate of the at least one recipient device	See ¶¶ D14-D48, D134-D140, D156-D161
[6.a.], [12.a.], [18.a.] receiving a message to test the authenticity of the at least one recipient computing device [apparatus], the generated message including a nonce; and	[5.b.] [10.b.] [15.b.] generating a message to test the authenticity of the at least one recipient device, the generated message including a nonce; [5.c.] [10.c.] [15.c.] sending the generated message to the at least one recipient device; and	

[6.b.], [12.b.], [18.b.] processing the generated message to indicate authenticity.	[5.d.] [10.d.] [15.d.] verifying if the at least one recipient device correctly processed the generated message.	See ¶¶ D14-D48, D134-D140, D156-D161
--	---	--------------------------------------

D281. It is my opinion that Kahn in view of Linn renders claims 6, 12, and 18 of the '956 patent obvious.

D282. It is my opinion that Kahn in view of Linn renders claims 5, 10, and 15 of the '007 patent obvious.

5. The Claims of the '576 Patent

a. Claim 18 of the '576 Patent

D283. Kahn and Linn describe methods and systems for distributing and rendering content. Kahn teaches a distributed system of repository servers for distributing digital objects (“*digital content*”) and terms and conditions (“*rights*”) to end-users. Ex. 1018 at 1:11-21, 5:61-6:5, 6:25-28, 6:39-43, 7:60-64, 8:20-30, 22:28-38; Figs. 1, 23-27. Content would be distributed in the SGML encoded envelope, envelope would be able to include the usage information fields (“*rights*”) described in both Kahn and Linn. These fields include the right to render, physically print, transfer (redistribute), perform, or modify content (“*said rights specifying how digital content can be rendered*”). Ex. 1018 at 7:17-22; Ex. 1019 at 15-16.

D284.Linn teaches rendering software installed on end-user's computer (“*apparatus*”) that renders content in accordance with terms and conditions attached to the content (“*rendering engine configured to render digital content*”). Ex. 1058 at 11, 13, 15-16, 20. An end-user has a computer that runs the Linn rendering software and is configured to retrieve content from the Kahn repository servers. Ex. 1018 (Kahn) at 23:48-26:38; Ex. 1058 at 14, 15-16; *id.* at 13-14 (the rendering software runs on computer systems and “*store[s] [information objects] . . . in standardized but **device-independent forms***”).

D285. The usage fields on each content envelope are enforced by the rendering software when the content is rendered (“*within said apparatus*”). Ex. 1019 at 11, 13, 15-16, 20. The rendering software checks the usage information associated with the object, and if the user has chosen a permitted use, rendering will be proceed in accordance with the use. Ex. 1058 at 13-14, 16. If the user has chosen an unpermitted use (*e.g.*, printing when the maximum number of copies has been made), “further processing should be inhibited” and the request will be denied (“*checking whether said request is for a permitted rendering . . .*”). *Id.* at 15-16.

D286. In the combined Kahn/Linn system, the content is encrypted as taught in Kahn. Ex. 1018 at 10:29-38. To decrypt the content, the rendering software would use the user's digital certificate to decrypt the content key stored in the envelope, decrypt the content, and display it in a manner appropriate for the user's

display or printer (“*processing the request to make said digital content available to the rendering engine for rendering*”). *Id.*; Ex. 1019 at 13-14, 15.

D287. The digitally signed authorization message from the Kahn payment server that is used in the Kahn/Linn system is an “*authorization object*.” Ex. 1018 at 23:23-45. The digitally signed authorization message is a digital work of a standard format, and it would be required both to purchase content and to use content (“*required to receive . . . and to use the digital content*”). Ex. 1018 at 23:23-45. Only the user who purchased the content and has a digitally signed authorization message can use the content because the rendering software will validate the user ID on the envelope against the authorization message before permitting rendering (*i.e.*, they are “*re[qu]ired to be included within the repository for rendering of the digital content*”). Ex. 1058 at 16.

D288. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'576 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
18. A method for controlling rendering of digital content on an apparatus having a rendering engine configured to render digital content and a storage for storing the digital content, said method comprising:	<u>“rendering engine”</u> Rendering software. <i>See ¶¶ D70-D77.</i> <u>“storage”</u> Rendering software runs on a computer with memory. <i>See ¶¶ D70-D77, D145-D155.</i>

<p>[18.a.] specifying rights within said apparatus for digital content stored in said storage, said rights specifying how digital content can be rendered;</p>	<p><u>“apparatus”</u> Rendering software runs on a computer. <i>See</i> ¶¶ D70-D77, D145-D155.</p> <p><u>“digital content stored in said storage”</u> Rendering software runs on a computer with memory. <i>See</i> ¶¶ D70-D77, D145-D155.</p> <p><u>“specifying rights ... specifying how digital content can be rendered”</u> Content is transmitted to the user in the <u>Linn</u> envelope, and it would have been obvious to add fields to envelope for the terms and conditions (<i>e.g.</i>, perform, use, modify, transmit) described in <u>Kahn</u>. ¶¶ D58-D62, D111-D114, D119-D129</p>
<p>[18.b.] storing digital content in said storage;</p>	<p>The computer system running the <u>Linn</u> rendering software would store content acquired from the <u>Kahn</u> repository servers in memory.</p>
<p>[18.c.] receiving a request for rendering of said digital content stored in the storage;</p>	<p><u>“request for rendering of said digital content stored in the storage”</u></p> <p>A user requests to render content using the rendering software. ¶¶ D70-D77.</p>
<p>[18.d.] checking whether said request is for a permitted rendering of said digital content in accordance with said rights specified within said apparatus;</p>	<p><i>See</i> ¶¶ D3, D5-D6, D9-D10, D55-D56, D59-D62, D70-D77, D104, D109, D119-D128</p>
<p>[18.e.] processing the request to make said digital content available to the rendering engine for rendering when said request is for a permitted rendering of said digital content;</p>	<p><u>“mak[ing] digital content available to the rendering engine”</u></p> <p>The rendering software will decrypt the</p>

	<p>content, and then render it.</p> <p>¶¶</p>
<p>[18.f.] authorizing a repository for making the digital content available for rendering, wherein the digital content can be made available for rendering only by an authorized repository, the repository performing the steps of:</p>	<p><u>“repository”</u></p> <p>Computer running rendering software</p> <p>In the combined system, the rendering software is a “repository” that can interact with the repository server.</p> <p>¶¶ D145-D155</p> <p>The <u>Kahn</u> repository server system.</p> <p>¶¶ D1-D4, D14-D21, D33-D48, D131-D144, D154-D155</p> <p><u>“authorizing a repository”</u></p> <p>User must have a digitally signed authorization message from a payment server</p> <p><u>“the digital content can be made available for rendering only by an authorized repository”</u></p> <p>The rendering software will not render content unless the user ID specified on the content envelope matches the user ID of the user</p> <p>¶¶ D76</p>
<p>[18.g.] making a request for an authorization object required to be included within the repository for rendering of the digital content; and</p>	<p><u>“authorization object”</u></p> <p>Digitally signed authorization message from the payment server</p> <p>See ¶¶ D33-D45, D156-D161</p>

[18.h.] receiving the authorization object when it is determined that the request should be granted.	Digitally signed authorization message from the payment server See ¶¶ D33-D45, D156-D161

D289. It is my opinion that Kahn in view of Linn renders claim 18 of the '576 patent obvious. If the claims of the Stefik patents are interpreted to require particular hardware components such as a tamper-resistant module or secure coprocessor, the claims of the '576 patent still are obvious in view of Kahn, Linn, and Dyad. See ¶¶ D163-D166, *above*.

b. Claim 1 of the '576 Patent

D290. Claim 1 of the '576 patent has several means-plus-function elements. I understand that the Board previously construed these elements in IPR2013-00139, Paper No. 15 (July 9, 2013) (“ZTE Inst. Dec.”). I have been instructed to apply the Board’s previous constructions in my analysis of the patentability of claim 1 of the '576 patent and its dependent claims.

D291. I also understand that ContentGuard has proposed claim constructions for these same elements in the *ContentGuard v. Apple* district court case that is ongoing in Texas. I understand that ContentGuard’s proposed claim constructions are a subset of the Board’s constructions, except for the element “means for requesting use of the digital content stored in the storage.” In my analysis of the patentability of claim 1 of the '576 patent and its dependent claims, I have

considered ContentGuard’s proposed claim constructions. Those proposed constructions do not alter my conclusions.

D292. The following table shows the Board’s previous constructions and ContentGuard’s proposed constructions.

Claim 1 Means-Plus-Function Element	Board’s structure	Patent Owner’s structure
“means for requesting use of the digital content stored in the storage”		Ex. 1009 at 16:35-44 (“user interface 1305”)
“means for processing a request from the means for requesting”	Ex. 1009 at 36:35-54, 37:9-26, 30:59-31:49, 30:20-21. <i>See</i> ZTE Inst. Dec. at 20-22.	Ex. 1009 at 36:35-40, 37:9-13
“means for checking whether the request is for a permitted rendering of the digital content in accordance with rights specified in the apparatus”	Ex. 1009 at 31:3-49. <i>See</i> ZTE Inst. Dec. at 22-23.	Ex. 1009 at 31:13-33.
“means for processing the request to make the digital content available to the rendering engine for rendering when the request is for a permitted rendering of the digital [content]”	Ex. 1009 at 33:3-49. <i>See</i> ZTE Inst. Dec. at 23-24.	Ex. 1009 at 33:3-49.
“means for authorizing the repository for making the digital content available for rendering”	Ex. 1009 at 41:52-42:16. <i>See</i> ZTE Inst. Dec. at 24-25.	Ex. 1009 at 41:52-42:16.
“means for making a [request] for an authorization object required to be included within the repository for the apparatus to render the digital content”	Ex. 1009 at 41:52-64. <i>See</i> ZTE Inst. Dec. at 25.	Ex. 1009 at 41:52-64.
“means for receiving the authorization object when it is determined that the request should be granted”	Ex. 1009 at 33:9-49. <i>See</i> ZTE Inst. Dec. at 26.	Ex. 1009 at 33:9-49.

D293. Claim 1 of the '576 patent recites “an apparatus for rendering digital” that performs two main functions—*rendering* content in accordance with rights, and *authorizing* the repository in the apparatus to be able to render content. As I explain in detail below, Linn and Shear show the *rendering* elements, while Kahn shows the *authorizing* elements. There are eleven elements of claim 1.

c. The Preamble of Claim 1 of the '576 Patent

D294. The first element of claim 1 is “An apparatus for rendering digital content in accordance with rights that are enforced by the apparatus, said apparatus comprising.” Linn and Shear each show this element.

D295. In Linn the “apparatus for rendering . . .” is a computer with “rendering software.” Linn explains that a computer with “standardized software” called “rendering software” is used to render digital content. *Id.* at 15 (“rendering of the information on a video display or printer are built-in functions of the rendering software”). Linn explains that the rendering software checks the usage information before permitting content to be transferred or rendered. *Id.* at 13 (“the required protection mechanisms [should be implemented] in computer systems as part of the application programs which deliver services to users”), 16 (“If a failure is detected when displaying or printing an object, further processing should be inhibited.”).

D296. In Shear the “apparatus for rendering . . .” is a computer with specialized software, that may have a hardware device plugged into it. Shear explains that a hardware device plugged into a computer, a storage medium such as a CD-ROM, a software program resident in the hardware device or computer, and “database access software” on the computer interoperate to store digital content, retrieve and decrypt it, and display and/or print it if various restrictions on use are satisfied. Shear at 16:57-62 (“The database access program resident in the host computer then controls the host computer to display and/or print the decrypted information. If desired, the program resident in the host computer 200 can prevent the user from doing anything other than displaying (and/ or printing) the decrypted data.”). Shear includes an alternate embodiment in which a single computer performs all of the functions, instead of having a computer and a hardware device plugged into the computer do so. Id. at 20:33-63, 21:15-27.

d. Second Element of '576 Claim 1: “a rendering engine”

D297. The second element of claim 1 is a “*rendering engine configured to render digital content.*” Linn and Shear each show this element.

D298. Linn shows “rendering software” that displays or prints digital content. Linn at 15.

D299.Shear shows a “database access program that “controls the host computer to display and/or print the decrypted information.” Ex. 1055 (Shear) at 16:57-62.

D300.A person of ordinary skill in the art would also have considered it obvious that an apparatus for rendering digital content would contain a rendering engine.

e. **Third Element of '576 Claim 1: “storage”**

D301.The third element of claim 1 is “*storage for storing the digital content.*” Linn and Shear each show this element.

D302.Linn shows use of a general purpose computer for storing envelopes containing digital content, and for running rendering software. Linn at 15.

D303.Shear shows a “storage device” such as a CD-ROM. Ex. 1055 (Shear) at Abstract (“One or more databases are encrypted and stored on a non-volatile *mass storage device*”), 9:1-4, 9:34-38 (“Predefined database(s) is (are) stored on *storage medium 100* in encrypted form”).

D304.A person of ordinary skill in the art would have considered it obvious that an apparatus for rendering digital content would contain a storage. For instance, computers with storages (e.g., hard disks) were commonly used to render digital content before the Stefik patents.

f. **Fourth Element of '576 Claim 1: “means for requesting use of the digital content stored in the storage”**

D305. The fourth element of claim 1 is “*means for requesting use of the digital content stored in the storage.*” ContentGuard construed this term as a “user interface.” Linn and Shear each show this element.

D306. Linn shows that users receive digital content in envelope, and can display or print (“*use*”) such content by interacting with (“*requesting use of*”) “built-in functions of the rendering software.” Linn at 15. Users interact with software via a user interface.

D307. Shear explains that “[t]o access database information, the user causes host computer 200 to execute software resident within it which permits the user to formulate a database access request (block 402).” Ex. 1055 (Shear) at 14:37-43, 14:58-15:5; *see also id.* at 5:2-10. Shear also explains that “Host computer 200 contains resident software and hardware which provides an interface for all database transactions.” *Id.* at 11:11-13. Thus, Shear has a user interface that allows a user to request use of the digital content stored in the storage.

D308. A person of ordinary skill in the art would have considered it obvious that an apparatus for rendering digital content would contain a user interface to permit the user to request use of the stored digital content. Users interact with software via a “user interface” almost by definition.

g. Fifth Element of '576 Claim 1: “a repository coupled to the rendering engine...”

D309. The fifth element of claim 1 is “a repository coupled to the rendering engine, wherein the repository includes.” As I explained above, Linn and Kahn show use of a “repository.” See ¶¶ D130-D155, *above*.

D310. In addition, Shear's system includes numerous security features that are fully compatible with use of repositories. The security features include a “tamper-proof decrypting device,” encryption, “mechanical and electronic safeguards,” security codes, and the like. *Id.* at Abstract, 3:63-66, 5:11-29, 8:6-12, 12:57-66. While Shear does not expressly disclose use of encrypted communications protocols, encrypted communication protocols were well known in 1994 and were conventionally used in applications desiring security. A person of ordinary skill in the art would thus have a motivation to, and could easily modify Shear to work together with Linn and Kahn using a “repository.”

h. Sixth Element of '576 Claim 1: “means for processing a request from the means for requesting.”

D311. The sixth element of claim 1 is “means for processing a request from the means for requesting.”

D312. This element has six parts. See ¶¶ B170-B178, *above*.

D313. Each of these six parts, as well as each part of each means-plus-function element in claim 1, describes a mundane process well-known to persons

of ordinary skill in the art. Each of the parts individually, and as a whole, are simple design choices, lacking any novel concepts, that could readily be implemented by persons of ordinary skill in the art with routine programming techniques. A person of ordinary skill in the art would not require a disclosure in the level of detail provided by the '576 patent as construed by the Board to implement these six parts.

D314. A person of ordinary skill in the art would also consider this claim element to be obvious in light of Shear.

D315. **First**, the claim construction requires that “processing begins when a requester repository has sent a message to **initiate a request**.” Analogously, Shear shows that a user causes the host computer to “formulate a **database access request**.” Ex. 1055 (Shear) at 14:40-43.

D316. **Second**, the claim construction requires that “the server repository checks the compatibility of the requester repository and the validity of the requester’s identification.” Shear shows coding the storage medium with unique passwords to ensure use of authorized decoder units (“compatibility of the requester repository”), and also shows that the system “may require the user to input identification and/or password information along with his access request” (“validity of the requester’s identification”). Ex. 1055 (Shear) at 15:3-9, 15:52-68.

D317. **Third**, the claim construction requires that “the requester and server repositories perform the common opening transaction steps of determining whether authorization is needed, and whether the requested transaction is permitted given the usage rights.” When a user attempts to access the storage medium, the system will determine whether a password is required, and if so, the system will require the user to input one (“determining whether authorization is needed”). Shear at 15:3-9, 15:52-68. Shear’s system allows the database owner and/or end-users to specify various restrictions on content usage, such as preventing a user from **“doing anything other than displaying (and/or printing) the decrypted data,”** restrictions on exceeding “a predetermined maximum (and/or minimum) usage limit,” and limits on duration, cost, and type of information which can be decrypted. *Id.* at 16:57-62, 18:10-12, 18:53-56, 19:7-11, 7:6-15.

D318. Kahn, Linn, and Shear each describes enforcing terms and conditions on how users can access content; a person of ordinary skill would understand that these enforcement processes are analogous. A person of ordinary skill in the art could incorporate using routine effort the enforcement mechanism the “terms and conditions” (“rights”) disclosed in Kahn or the control restrictions disclosed by Linn into the Shear system. Thus, Kahn and Linn in view of Shear renders obvious “determining whether the requested transaction is permitted given the usage rights.”

D319. **Fourth**, the claim construction requires that the “requester and server repositories perform a transmission protocol to read and write blocks of data, and then the requester repository renders the digital work.” Shear shows that “[i]n response to the user’s access request, host computer 200 accesses index portion 102 stored on medium 100 and obtains from the index portion the addresses of (or index keys corresponding to) each block 106 of the encrypted database which satisfies the user’s access request (block 406) (index portion decryption is performed at this time if necessary). **Host computer 200 then reads the appropriate block(s) 106 of the encrypted database from storage medium 100 and stores these blocks of information into its own internal random access memory** (block 408).” Ex. 1055 (Shear) at 14:60-15:2. The database access program in Shear then renders the digital content. Ex. 1055 (Shear) at 16:57-59 (“The database access program resident in the host computer then controls the host computer to display and/or print the decrypted information.”).

D320. **Fifth**, the claim construction requires that the “the contents are removed from the rendering device and the requester repository.” Shear shows that, optionally, “[t]he host computer may decrypt database information ‘on the fly’ and not retain much encrypted or decrypted information in memory at any one time to help prevent copying.” Shear at 21:35-38. A person of ordinary skill in the

art would understand that decrypting information “on the fly” entails sequentially removing (“not retain[ing]”) the information from memory.

D321. **Sixth**, the claim construction requires that “the requester and server repositories perform the common closing transaction steps of updating the usage rights and billing information.” Shear is generally related to metering usage of digital content, and billing users for that usage, and these elements are discussed throughout the specification. See, e.g., Ex. 1055 (Shear) at Abstract, 17:1-8 (describing metering database usage), 17:36-48 (describing storing billing log entries in memory).

i. **Seventh Element of ’576 Claim 1: “means for checking whether the request is for a permitted rendering...”**

D322. The seventh element of claim 1 is “means for checking whether the request is for a permitted rendering of the digital content in accordance with rights specified in the apparatus.”

D323. This element has four parts. See ¶¶ B170-B178, *above*.

D324. A person of ordinary skill in the art would have considered each part and this element as a whole to be obvious for the reasons I explained above. See *supra* ¶ D313.

D325. A person of ordinary skill in the art would also consider this claim element to be obvious in light of Kahn, Linn, and Shear.

D326. **First**, the claim construction requires that “the requester repository determines whether an authorization certificate or a digital ticket is needed.” As I explain further below in the discussing of the authorizing function of the rendering apparatus, Kahn describes an initial authorization process whereby “before a user can retrieve any objects for which payment is required, the user must first establish an account with a payment server system 702 on the network.” Ex. 1018 (Kahn) at 22:29-31. A person of ordinary skill in the art would consider it obvious that to make Kahn’s authorization process effective, the rendering apparatus would need to determine whether it needs to obtain an authorization object (“certificate or digital ticket”) and could not simply render content regardless of whether or not it was authorized.

D327. Shear also describes an authorization process based on the user’s ID and/or codes in the storage medium. Ex. 1055 (Shear) at 15:3-13, 15:52-68. Shear’s authorization process could easily incorporate an additional check for the presence of a valid “received signed message” as described by Kahn. A person of ordinary skill in the art would understand how to incorporate this additional check using routine programming techniques.

D328. **Second**, the claim construction requires that “the server repository generates a transaction identifier.” Shear shows that “if decoder control logic 316 of decoder/biller 300 grants authority to proceed (block 412), the decoder control

logic begins a ‘billing cycle’, and stores information logging the billing cycle into non-volatile memory 314 (block 416). . . . The information stored in non-volatile memory 314 may thus be used to create an ‘audit trail’ which tracks different users (or groups of users) and their database usages.” Ex. 1055 (Shear) at 15:22-51. A person of ordinary skill in the art would have considered it obvious that the “information logging the billing cycle” or the “audit trail” tracking database usages would include, or at a minimum could include, a “transaction identifier” to aid in generating detailed bills and tracking unauthorized use, as taught by Shear. Shear at 15:33-51.

D329. **Third**, the claim construction requires that “the server repository determines whether the right is granted and whether time, security, and access based conditions are satisfied.” Shear explains that its system “may include an enabling/disabling mechanism which prevents a user from accessing the stored database information if he fails to pay his bill,” if he “exceeds a predetermined maximum (and/or minimum) usage limit,” if “the real time clock ever ceases to operate,” if a “preset real time deadline” has passed, or if certain user-defined parameters have been set to “limit[] the user’s own use of database information.” Ex. 1055 (Shear) at 18:10-12, 18:25-27, 18:38-19:11. These limitations are “time, security, and access based conditions.”

D330. **Fourth**, the claim construction requires that “the server repository determines whether there are sufficient copies of the work to distribute.” Shear discloses that “[a] permanent record of the blocks (records or other subdivisions) which have been accessed may be retained in non-volatile memory 314 so that the user can be prevented from copying an excessive amount or selected database properties or segments over a period determined by the database owner.” Ex. 1055 (Shear) at 19:1-7. The above description in Shear is analogous to determining if there are sufficient copies of the work to distribute.

D331. A person of ordinary skill in the art would consider the step of determining whether there are sufficient copies obvious based on this disclosure. A person of ordinary skill in the art would also consider this step obvious because it is a classic function of a library system. A person of ordinary skill in the art would also consider this step obvious based on the disclosure in Linn of “copy counters” that enforce “limited redistribution” of digital works. Linn at 16.

j. Eighth Element of '576 Claim 1: “means for processing the request to make the digital content available...”

D332. The eighth element of claim 1 is “means for processing the request to make the digital content available to the rendering engine for rendering when the request is for a permitted rendering of the digital [content].”

D333. This element has three parts. *See ¶¶ B170-B178, above.*

D334. A person of ordinary skill in the art would have considered each part and this element as a whole to be obvious for the reasons I explained above. See *supra* ¶ D313.

D335. A person of ordinary skill in the art would also consider this claim element to be obvious in light of Kahn, Linn, and Shear.

D336. **First**, the claim construction requires that the “transaction information is provided to the server repository by the requester repository.” Shear explains that a user “**inputs an access request** (block 404) using a keyboard or other standard I/O device connected to host computer 200” (“transaction information”). Ex. 1055 (Shear) at 14:58-62. Shear explains that “[i]n response to the user’s access request, host computer 200 [(“requestor repository”)] accesses index portion 102 stored on medium 100 [(“server repository”)].” Ex. 1055 (Shear) at 14:58-62. In doing so, the host computer necessarily transmits information from the access request to the storage medium, including information identifying the requested content.

D337. **Second**, the claim construction requires that the “the server repository transmits a block of data to the requester repository and waits for an acknowledgement provided by the requester when the block of data has been received. Unless a communications failure terminates the transaction, that process repeats until there are no more blocks to transmit.” Shear discloses that:

In response to the user's access request, host computer 200 accesses index portion 102 stored on medium 100 and obtains from the index portion the addresses of (or index keys corresponding to) each block 106 of the encrypted database which satisfies the user's access request (block 406) (index portion decryption is performed at this time if necessary). *Host computer 200 then reads the appropriate block(s) 106 of the encrypted database from storage medium 100 and stores these blocks of information into its own internal random access memory* (block 408)."

Ex. 1055 (Shear) at 14:60-15:2.

D338. In the preferred embodiment Shear uses sequential transmission of "fixed-length blocks" to accomplish this data transfer, just like the '576 patent.

Ex. 1055 (Shear) at 16:42-56.

D339. While Shear does not expressly disclose waiting for an acknowledgement provided by the requester, such a step is inherent in a scheme for sequential transmission because otherwise in the case of a transmission error the sender would continue sending data and waste transmission resources.

D340. In addition, the exact transmission protocol for data blocks, including whether to wait for an acknowledgment after transmitting each block of data, would have amounted to a design choice for a person of ordinary skill in the art.

Ex. 1055 (Shear) at 16:42-44 ("The specific steps performed to decrypt data . . . depend on the particular data encryption/decryption scheme used.").

D341. Many known transmission protocols existed in 1994, including protocols that used per-block acknowledgments. Thus, the foregoing transmission protocol would have been considered obvious to a person of ordinary skill in the art.

D342. *Third*, the claim construction requires that the “*the requester repository commits the transaction and sends an acknowledgement to the server repository.*” As I explained immediately above, Shear shows that the host computer reads the appropriate blocks from the content database and stores them in its own internal memory, making them available for rendering (“*commits the transaction*”). Ex. 1055 (Shear) at 14:60-15:2, 16:57-62.

D343. A person of ordinary skill in the art would have considered sending an acknowledgment to the server repository to be an obvious design choice. Such acknowledgment messages indicating successful completion of a transaction between two computers (or processes in a single computer) were common and well-known techniques in computer programming.

k. Ninth Element of '576 Claim 1: “means for authorizing the repository for making the digital content available for rendering...”

D344. The ninth element of claim 1 is “means for authorizing the repository for making the digital content available for rendering, wherein the digital content can be made available for rendering only by an authorized repository, the

repository comprising.” This element and the subsequent means-plus-function elements of claim 1 relate to authorizing a repository.

D345. This element has four parts. *See* ¶¶ B170-B178, *above*.

D346. Kahn discloses an authorization process that renders this element obvious.

D347. **First**, the claim construction requires that “a communication channel is set up between the server and the remote repository.” Kahn explains that “[b]efore a user can retrieve any objects for which payment is required, the user must first establish an account with a payment server system 702 on the network.” Kahn at 22:29-31.

D348. To establish an account, “[t]he user (or his software agent formats (706) a setup-new-account message containing the following . . . the signed message is sent (708) to the payment server.” Ex. 1018 (Kahn) at 22:49-50. Setting up a communication channel is an inherent, mundane, and obvious prerequisite to sending the new account message from the user to the payment server.

D349. **Second**, the claim construction requires that “the server repository performs a registration process with the remote repository.” As described immediately above, Kahn describes a process by which a user registers his system (“server repository”) with a remote payment server (“remote repository”). *Id.* at 22:29-31, 22:49-50.

D350. **Third**, the claim construction requires that “the server repository invokes a ‘play’ transaction to acquire the authorization object.” Kahn’s authorization process renders this item obvious by allowing the server repository to acquire a signed formatted message (“authorization object”). A person of ordinary skill in the art would understand how to acquire digital objects from a remote location using ordinary data transfer techniques, such as transmission of an encrypted message. I understand that the functionality of this process is controlling in assessing patentability. I understand that the particular label of a “play” transaction as compared to the label of a “setup-new-account” transaction of Kahn is not controlling.

D351. **Fourth**, the claim construction requires that “the server repository performs tests on the authorization object or executes a script before signaling that authorization has been granted for rendering content.” Kahn’s authorization process inherently discloses this item, because it discloses that the payment server “signs” the formatted message. Ex. 1018 (Kahn) at 23:39-41. Signing implies that the recipient can “test” the received message to verify that the message came from a valid source, in accordance with standard techniques for digital signatures that were well known in 1994. A person of ordinary skill in the art would also have considered it obvious to “test” the received message to ensure it was valid. Such

validity checks on received data were common and well-known to persons of ordinary skill in the art.

l. Tenth Element of '576 Claim 1: “means for making a request for an authorization object...”

D352. The tenth element of claim 1 is “means for making a request for an authorization object required to be included within the repository for the apparatus to render the digital content.”

D353. This element has four parts. *See ¶¶ B170-B178, above.*

D354. Each part is duplicative of the previous element (“ninth element of claim 1”). Therefore, my analysis of the ninth element of claim 1 applies to the tenth element too.

m. Eleventh Element of '576 Claim 1: “means for receiving the authorization object when it is determined that the request should be granted.”

D355. The eleventh element of claim 1 is “means for receiving the authorization object when it is determined that the request should be granted.”

D356. The Board previously construed this element to describe a “particular transmission protocol” for sending an authorization object from a remote repository to a server repository. *See ¶¶ B170-B178, above.*

D357. The details of this transmission protocol are the same as the transmission protocol for sending blocks of data from a server repository to a requester repository. I explained above (“eighth element of claim 1”) the reasons

why a person of ordinary skill in the art would have considered such a transmission protocol to be obvious. Therefore, my analysis of the eighth element of claim 1 applies to the eleventh element too.

D358. It is my opinion that Kahn and Linn in view of Shear renders claim 1 of the '576 patent obvious.

6. Claims 2 and 19 of the '576 Patent

D359. Claim 2 recites “[t]he apparatus as recited in claim 1, wherein the means for checking comprises means for comparing a requested use with a use specified by the rights.”

D360. As I explained above, Kahn and Linn disclose “usage rights” that are enforced by rendering software (*e.g.*, checking if display/performing object is permitted or if printing is permitted) on the end-user device. *See* ¶¶ D119-D129.

D361. Shear likewise discloses a variety of restrictions that are enforced by the end-user device, including restrictions on manner of use (*e.g.*, displaying and/or printing). *See* ¶¶ D91-D93. Enforcing such restrictions necessarily entails comparing a requested use with a use specified by the rights.

D362. This claim element also appears redundant with the element of claim 1 “means for checking whether the request is for a permitted rendering of the digital content in accordance with rights specified in the apparatus.”

7. Claims 4 and 21 of the '576 Patent

D363. Claims 4 and 21 recite “requesting a transfer of the digital content from an external memory to the storage.”

D364. Shear’s preferred embodiment involves use of an external “storage medium” such as an optical disk or CD-ROM. Ex. 1055 (Shear) at Abstract, 9:34-64. Shear discloses that a digital content database “may be loaded onto storage medium 100 in a conventional fashion,” such as by duplicating a “master” medium. Ex. 1055 (Shear) at 10:10-20. These “loading” and “duplicating” processes comprise transferring digital content from an external memory (*e.g.*, the master medium) to the storage (*i.e.*, the storage medium 100).

D365. Shear also describes optional use of a “direct connect” mode of operation in which “the bulk of a database can be stored on and accessed locally from a local storage medium 100,” but a “remote centralized facility” can be “accessed via a telecommunications link to provide extremely current information . . .” Shear at 19:27-20:3. Based on this disclosure, a person of ordinary skill in the art would have considered it obvious that the storage medium 100 could be re-loaded periodically by transferring information from computer memory located a centralized facility (“external memory”).

D366. In the combined system, an end-user has a computer that runs the Linn rendering software and is configured to retrieve content from the Kahn

repository servers. Ex. 1018 at 23:48-26:38; Ex. 1019 at 14, 15-16. The content is stored in memory (“*external memory*”) on the repository servers, and the rendering software requests the servers to transfer the content (“*requesting a transfer*”) to the end user’s computer for storage (“*storage*”). Ex. 1058 at 13-14 (the rendering software runs on computer systems and “*store[s]* [information objects] . . . in standardized but *device-independent forms*.”).

D367. In addition, a person of ordinary skill would have found it obvious to configure the rendering software to permit an end-user to back-up content onto external storage, such as a magnetic or optical disk.

8. Claims 7 and 24 of the '576 Patent

D368. Claims 7 and 24 recite “wherein the digital content is video content.”

D369. Kahn shows use of video content. Ex. 1018 (Kahn) at 22:59-60 (“Optional category of use (e.g., this account is used to retrieve video objects only.”), 6:4 (“e.g., a video game”).

D370. Shear focuses on textual material, but its system is designed to work with any type of digital content that can be loaded on storage media such as CD-ROM disks. Ex. 1055 (Shear) at 9:45-64. A person of ordinary skill in the art would understand that video content could be used with Shear.

9. Claims 8 and 25 of '576

D371. Claims 8 and 25 recite “wherein the apparatus is a portable device.”

D372.Linn describes a system configurable to work with any apparatus that can run “rendering software” and a display. Ex. 1058 (Linn) at 14-15.

D373.Shear likewise describes a system configurable to work with any apparatus that can connect to the hardware plug-in device, can run software, and has a display. Ex. 1055 (Shear) at Abstract, 9:26-33.

D374. A person of ordinary skill in the art would have considered it obvious that the systems of either Linn or Shear could be used with a portable laptop computer.

10. Claims 9 and 26 of the '576 Patent

D375. Claims 9 and 26 recite “wherein the rights are provided by a provider of the digital content.”

D376.Kahn shows that the rights in the form of “terms and conditions” are ordinarily provided by the same “rights holder” that places the digital content object into the repository. Ex. 1018 (Kahn) at 3:4-17 (“Another general aspect of the invention concerns enabling holders of rights in digital objects to control terms and conditions under which rights in the digital objects may be granted to others.”), 15:8-53 (describing a rights holder placing an object into a repository and setting terms and conditions).

D377.Linn explains that “publishers and authors” can use “rendering software” to “creat[e] an original information object” in the form of digital content together with terms and conditions structured in an electronic envelop. Linn at 15.

D378.Shear explains that database owner and/or end-users can specify various restrictions on content usage. *Id.* at 16:57-62, 18:10-12, 18:53-56, 19:7-11.

11. Claims 10 and 27 of the ‘576 Patent

D379.Claims 10 and 27 recite “wherein the means for processing comprises transmitting the digital content to the rendering engine.”

D380.In the Kahn/Linn system, when the user requests to render content consistent with the usage information, the rendering software will decrypt the content and render it in a form suitable for the end user’s computer. Ex. 1058 (Linn) at 15, 16; Ex. 1018 (Kahn) at 10:29-38. A person of ordinary skill would understand that this requires the content to be transmitted from memory to the processor.

D381.As I explained above, Shear explains that the database with digital content transmits the digital content to the host computer (containing the database access program that renders the content). *See* ¶¶ D305-D308 (fourth element, “requester and server repositories perform a transmission protocol to read and write blocks of data”).

D382. Transmitting the digital content to the rendering engine is also an obvious, necessary pre-requisite to rendering the digital content. The rendering engine cannot render content that it does not have.

12. Claims 15 and 32 of the '576 Patent

D383. Claims 15 and 32 recite “wherein the rights are embodied in software instructions which implement the use privileges for the rights.”

D384. Each of Kahn, Linn, and Shear show use of software to interpret and enforce the end-user’s rights to access and use the digital content. See ¶¶ D1-D20, D70-D77 (Kahn and Linn disclose “usage rights”); Ex. 1055 (Shear) at Abstract, 14:40-43, 16:57-62.

D385. A person of ordinary skill in the art would also have considered it obvious that software would be used to implement usage rights. Usage rights are provided in digital form.

13. Claim 29 of the ‘576 Patent

D386. Claim 29 depends from claim 18 and species “wherein the rendering engine is configured to render digital content into a medium that is not further protected by rights and the request is for rendering the digital content into a medium that is not further protected by rights.”

D387.Linn shows that content can be physically printed onto paper. *E.g.*, Ex. 1058 (Linn) at 15. Copies on paper are in a medium that is not protected by the system.

14. Claim 34 of the ‘576 Patent

D388.Claim 34 recites the method of claim 18, “further comprises: requesting receipt of digital content stored externally; and receiving the digital content if it is permitted to receive the digital content.”

D389.As explained with respect to claim 21, in the Kahn/Linn system, an end-user has a computer that runs rendering software and is configured to retrieve content from the Kahn repository servers. Ex. 1018 at 23:48-26:38; Ex. 1019 at 14, 15-16.

D390.The repository servers will transfer the content to the rendering software only if the user is authorized (*e.g.*, has a valid digitally signed authorization message that permits the transfer), and therefore, the rendering software will receive the content only “*if it is permitted to receive the digital content*”.

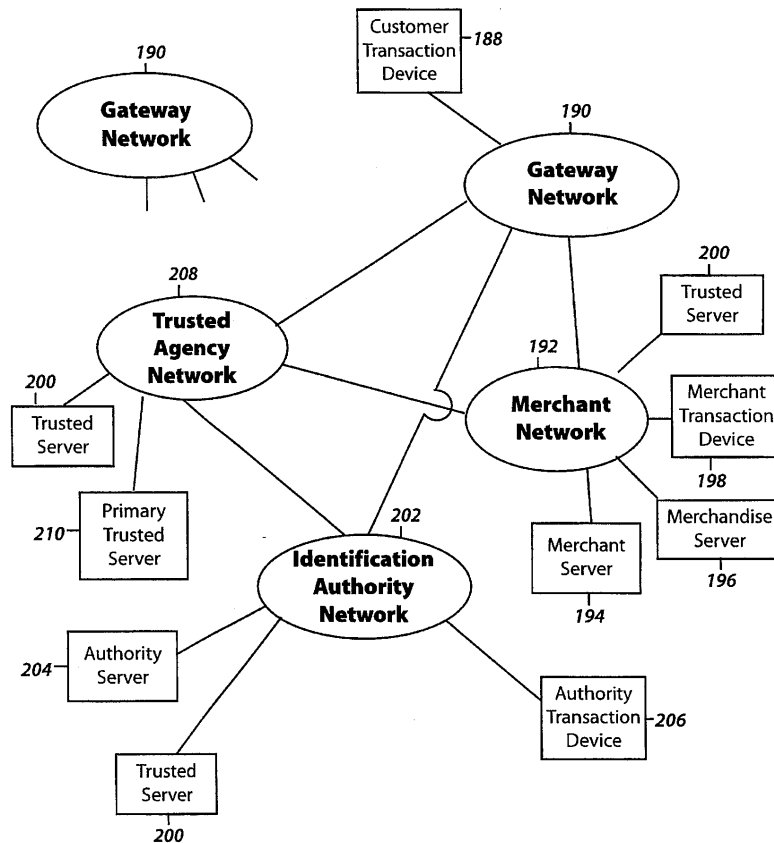
VIII. SECTION E: Rosen (Exhibit 1020) in View of Hartrick (Exhibit 1021)

A. Overview of Rosen (Ex. 1020)

E1. Rosen describes a method for securely distributing media (*e.g.*, software, movies, and other electronic content). Ex. 1020 (Rosen) at 1:5-11, 4:55-57. As Rosen explains:

The present invention relates to a system for facilitating open electronic commerce. In particular, the system utilizes tamper-proof electronic units, referred to as ‘trusted agents’ in combination with money modules to create a secure transaction environment for both the buyer and seller of electronic merchandise and services.

Ex. 1020 (Rosen) at 1:5-11. Rosen shows a distributed network of merchant computers that customers can access, search for content, and then purchase it. This is depicted in Figure 5:



E2. Rosen explains that content is exchanged using “transaction devices,” which could be a personal computer, a portable computer, or other device. *Id.* at 8:28-34, 25:63-26:39.

E3. Each Rosen transaction device includes a “trusted agent,” which is a tamper-resistant module that enforces secure protocols used to distribute and access content. *Id.* at 4:7-13 (“[T]he present invention introduces trusted agents 2, 4 for both the customer and merchant. **A trusted agent** is a combination of hardware and software components. **It is tamperproof and contains secure protocols . . .**”); *id.* at 4:27-32 (“ . . . In order **to guarantee the behavior of a**

trusted agent, the protocols are physically protected. Thus neither party can modify the protocols to the disadvantage of the other party.”); *id.* at 1:5-11, 7:65-8:7. I note that while Rosen describes the modules as “tamperproof,” the more accurate term is “tamper-resistant.”

E4. Rosen also employs a network of certificate authorities to authenticate trusted devices. Each trusted agent has a digital certificate issued by a trusted certificate authority, and this certificate includes a unique identification number and a public key of the trusted agent. *Id.* at 11:6-12:25.

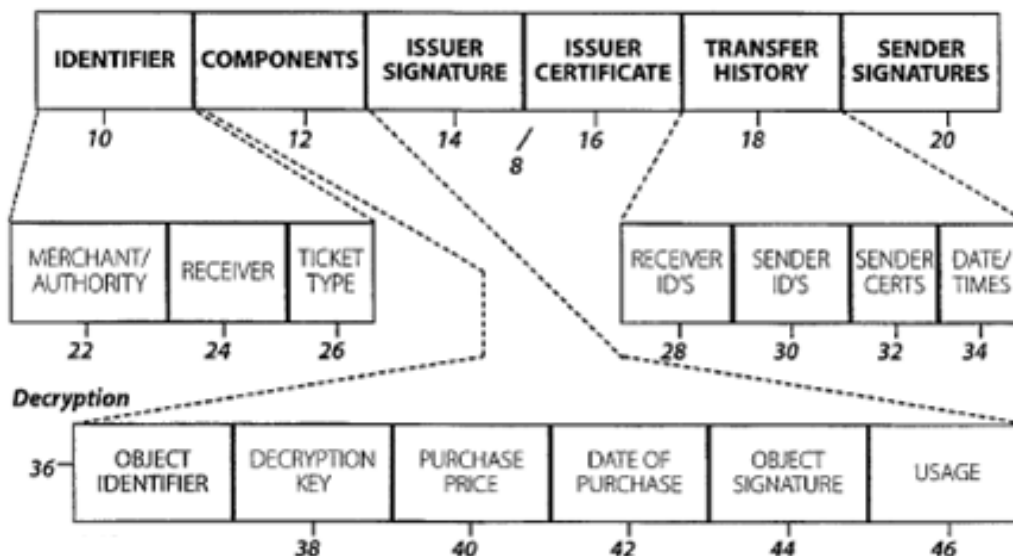
E5. Rosen explains there is a hierarchy of trusted entities including primary trusted servers, trusted servers, and trusted agents. Ex. 1020 (Rosen) at 10:50-55 and Fig. 6A. The root of trust in the system resides in a “Trusted Agency,” which plays a crucial role in component fabrication, initialization, and ongoing security operations. Ex. 1020 (Rosen) at 11: 5-13, 12: 16-21.

E6. Rosen explains that digital content is contained in an “electronic object.” Ex. 1020 (Rosen) at 4:41-45 (“Electronic merchandise is any goods that can be represented in electronic form, and in the preferred embodiment described herein consists of either a ticket or an encrypted electronic object (EO) and its associated decryption ticket.”).

E7. Electronic objects are encrypted, and they are always associated with a decryption “ticket” that can be used to decrypt and use the media. Ex. 1020

(Rosen) at 4:54-55 (“A decryption ticket is always associated with a particular encrypted electronic object”), 5:59-6:13, 4:55-5:3 (“The transferred electronic object is cryptographically secure from inspection and use by the receiving customer or any other third party unless they have access to the decryption ticket.”).

E8. A decryption ticket contains an object identifier for associating with a particular object, a decryption key, and a “Usage” field that specify “restrictions on usage of the electronic object.” Ex. 1020 (Rosen) at 5:59-6:13, Fig. 2. A person of ordinary skill would understand that the “Usage” field should be used to store usage rights, as was conventional. *See* ¶¶ A91-A97, *above*. An example of a decryption ticket is shown in Figure 2, reproduced in relevant part below:



E9. Rosen explains that the tamper-resistant module stores decryption tickets, and after a user has paid for the content, these can be retrieved when

content is to be executed or used. Ex. 1020 (Rosen) at 9:60-61 (“A Ticket Holder function 148 creates tickets 8 in a MTA 4 [Merchant Trusted Agent] or **stores and retrieves tickets** 8 in a CTA 2 [Customer Trusted Agent].” (emphasis added)), 4:48-52 (“Tickets may be thought of as the property of the trusted agents”). A ticket for an object is retrieved by passing its object identifier to the trusted agent, which then uses the identifier to locate the corresponding ticket. *Id.*; *id.* at 24:52-55.

E10. Rosen shows that electronic objects are stored in encrypted form. Ex. 1020 (Rosen) at 23:17-22 (“the electronic object is encrypted so that it may be stored outside of the trusted agent”), 4:54-55 (“A decryption ticket is always associated with a particular encrypted electronic object”).

1. Transaction Devices

E11. As I have explained, each transaction device in the Rosen system contains a trusted agent. Ex. 1020 (Rosen) at 2:12-14, 4:4-14, 7:65-8:2, Figs. 4A-4D. The trusted agent is used to facilitate secure transactions between devices. *Id.* at 4:4-39, 11:6-24; *see id.* at 8:56-10:2. Each transaction device also includes a processor and a money module. 7:65-8:2.

E12. Rosen explains that the trusted agent contains a tamper-resistant module, and therefore, is physically secure. *Id.* at 11:6-30 (“The Trusted Agency **guarantees the protocols and physical protection of each trusted agent** 120 in

the system. Trusted agents 120 are manufactured in a physically secure environment under control of the Trusted Agency. The components are fabricated, assembled, and loaded with software in this environment. The trusted agents 120 are then tamper-proofed and can only be communicated with through their external interface.”). Because the trusted agents are physically secure, users of the system can trust that every agent will behave as expected. *Id.* at 4:27-32 (“ . . . In order to **guarantee the behavior of a trusted agent, the protocols are physically protected.** Thus neither party can modify the protocols to the disadvantage of the other party.”).

E13. Rosen also explains that the trusted agents guarantee the “protocols” used the by system, which means that the trusted agents ensure an electronic object can be used only after a customer has paid and complied with any other conditions.

E14. The standard components of a trusted agent are shown in Figure 4A:

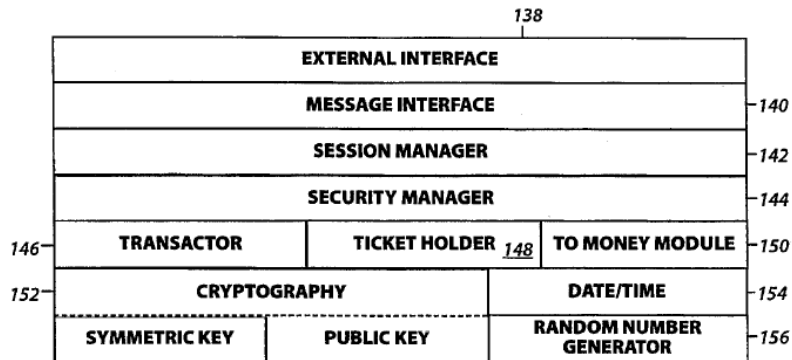


Figure 4A

E15. Rosen explains that the Ticket Holder is used to store decryption tickets in a customer trusted agent. Ex. 1020 (Rosen) at 9:60-61 (“A Ticket Holder function 148 creates tickets 8 in a MTA 4 [Merchant Trusted Agent] or stores and retrieves tickets 8 in a CTA 2 [Customer Trusted Agent].”), 19:37-39 (“If the purchaser accepts the merchandise, then Purchase A sends the decryption ticket to Ticket Holder A for storage (steps 490-492).”).

E16. Rosen also shows that each transaction device contains a “processor,” which means it computerized device (*e.g.*, a personal computer). Rosen at 7:65-8:2.

E17. Rosen explains that many different devices can be or incorporate a transaction device. For example, a transaction device can be a personal computer, or it can be a special unit that is incorporated into the computer. Ex. 1020 (Rosen) at 25:67-26:7, 8:22-34. Rosen explains each transaction device has a “Human/Machine Interface function” that “provides the look and feel of the transaction device” and can “include a keyboard, mouse, pen, voice, touch screen, icons, menus, etc.” Ex. 1020 at 8:35-38.

E18. Rosen also shows that a transaction device could be a portable unit. Ex. 1020 (Rosen) at 7:13-17, 26:1-3.

E19. Rosen explains that a transaction device can be used to execute or view content in an electronic object, but only if the device’s trusted agent has the

associated decryption ticket. Ex. 1020 (Rosen) at 4:65-5:3 (“*The transferred electronic object is cryptographically secure from inspection and use* by the receiving customer or any other third party *unless they have access to the decryption ticket*. The decryption ticket, in turn, is the ‘*property*’ of the CTA and may only be used upon successful completion of the purchase transaction.”), 8:28-34, 24:52-55. *id.* at 24:52-55 (“[A]fter payment the Open Merchandise subroutine is called (step 434). Referring to FIG. 22, Purchase A checks if merchandise is an electronic object (steps 736-738). If so, Ticket Holder A sends the decryption key and the electronic object identifier from the decryption ticket to a **host transaction application for its use in decryption of the electronic object** (steps 740-742).”).

E20. To render content, a transaction device has “transaction applications”, which are software programs. *Id.* at 7:65-8:7, 8:22-34 (“Transaction Applications 130 may perform a variety of tasks. . . [A] transaction application may provide for the interim storage of electronic objects and possibly execute objects. In order to execute an electronic object, there may be additional object processors depending on the type of electronic object (e.g., movie, book, multimedia game, etc.) . . .”). Requests placed through the Human/Machine Interface are relayed by a Message Manager, that can, among other tasks , route the message requesting execution of stored electronic content (“*requesting use of the digital content*”) to relevant the transaction application. *See id.* at 8:26-34, 8:38-43, 8:49-52. In other words, in

order for the application to render content for a user, the transaction device would first receive a request from a user to render that content. *E.g.*, Ex. 1020 (Rosen) at 8:35-40 (describing the “Human/Machine Interface”).

2. Electronic Objects & Tickets

E21. Rosen shows each electronic object has a header and a body. Ex. 1020 (Rosen) at 5:59-66 (“. . . Electronic objects themselves (e.g., movies) are comprised of a header and a body. **The header contains an object identifier which ties to the object identifier 36 in the decryption ticket.**”). The header is unencrypted and contains descriptive information, while the body remains encrypted. *Id.* at 5:59-6:2, 19:20-30. It also includes an identifier that ties it to the decryption ticket.

E22. Rosen explains that an electronic object is permanently associated with a decryption ticket. *Id.* at 4:54-5:3 (“A decryption ticket is always associated with a particular encrypted electronic object. Examples of electronic objects are computer software, games, movies, or information products like electronic newspapers and books. . . An encrypted electronic object can be decrypted by unique information in its associated decryption ticket. **Together, the encrypted electronic object and its decryption ticket comprise the electronic merchandise transferred by the merchant.** The transferred electronic object is

cryptographically secure from inspection and use by the receiving customer or any other third party unless they have access to the decryption ticket. . .”).

E23. Rosen shows that each ticket contains a digital signature of the electronic object. Ex. 1020 (Rosen) at 6:7-12. The signature is “used to detect if a signed electronic object has been altered in any way since the time it was signed” and thus verify its integrity. *Id.* at 6:7-12 (“An Object Signature field 44 contains a digital signature of the electronic object. Digital signatures are well known in the art and are used to detect if a signed electronic object has been altered in any way since the time it was signed. Thus, electronic object integrity may be checked.”); *id.* at 5:38-52, 19:13-23.

E24. The ticket also contains the digital certificate of the entity that created the ticket, which can be used in the validation process. *Id.* at 7:30-41, Fig. 2 (items 16 and 32); *id.* at 7:30-41; *id.* at 30:26-34. If the ticket was transferred after it was created, it also will include a digital signature and certificate of any entity that to which the object was transferred. *Id.* at 7:42-61; *id.* at 26:25-39.

E25. When content is exchanged, a trusted agent will validate the integrity of an electronic object using the digital signature and digital certificate. *Id.* at 6:7-12, 19:13-23. If the validation fails, the trusted agent will abort the transaction, and delete the content. *Id.* at 19:13-23, 14:22-37.

E26. Rosen shows that content is stored in encrypted form. Ex. 1020 (Rosen) at 23:17-22 (“the electronic object is encrypted so that it may be stored outside of the trusted agent”); *id.* at 4:65-5:3. For example, Rosen shows that if a consumer is unhappy with a purchase, the user can return an electronic object to the merchant. When the customer returns the object, it is still in encrypted form. *Id.* at 30:30-34.

E27. In addition to decryption tickets, Rosen describes other types of “tickets” that are used for different purposes and are not associated with electronic objects. *Id.* at 4:52-5:23. For example, “[a] credential ticket [that] identifies the ‘owner’ and permits specific privileges.” *Id.* at 5:4-5 “, 5:34-35, 6:14-25. Rosen explains:

All [merchant transaction devices] contain a credential identifying the owner/merchant (e.g., NYNEX, Ticketron, etc.). Such merchant credentials may, for example, be issued by a merchant identification authority controlled by the Trusted Agency. On the other hand, customer credentials held by [customer transaction devices] may include driver’s licenses or credit cards issued by various identification authorities.

Id. at 18:5-13, 5:5-7, 26:44-54. These tickets are used as proof of authorization in various contexts, such as during a purchase transaction. *Id.* at 18:5-34 (merchant credentials and checking credentials generally), 19:40-49.

3. Certificate Authority

E28. Rosen explains that devices are certified by a trusted certificate authority, whose root of trust is a Trusted Agency. The trusted agency delegates trust in a hierarchical fashion to primary trusted servers, trusted servers, and trusted agents. Ex. 1020 (Rosen) at 10:40-11:38. Rosen describes the basic functions of the Trusted Agency as follows:

With reference to FIG. 5, security for the open electronic commerce system is provided by a network of trusted servers 200 situated at a Trusted Agency Network 208, at merchant networks 192, and at identification authority networks 202. The trusted servers 200 are tamper-proof processors that perform four primary functions: certification of trusted agents 120, distribution of untrusted lists, distribution of primary trusted server public key lists, and resolution of customer/merchant disputes.

Id. at 10:41-50.

E29. To participate in the Rosen system, a device must have a valid certificate. Each certificate binds a public key of a device to the device's unique identification number. Devices obtain certificates during an initialization process. Ex. 1020 (Rosen) at 11: 5-13, 12:16-21.

E30. When a trusted agent is initialized, it generates a (public-key, private-key) pair and passes the public key to a trusted server. Ex. 1020 (Rosen) at 11:13-30. The trusted server assigns the module a unique identification number, and creates a digital certificate with the module's identification number and public key,

and passes this certificate to the module. The certificate allows the module to be identified as a trusted agent.

E31. Similarly, each trusted server obtains a unique identification number and public-key certificate for itself by engaging in an initialization protocol with a primary trusted server. Ex. 1020 (Rosen) at 11:31-38.

E32. Each certificate contains the agent's identification number, public key, an expiration date, and the digital signature of the Trusted Agency. Ex. 1020 (Rosen) at 11:39-60. The certificate can be validated using the standard certificate validation process. *Id.* at 11:61-12:15; *see* ¶¶ A144-A150, *above*.

E33. The Trusted Agency is charged with several ongoing security operations. One ongoing security operation of the Trusted Agency includes distributing updated lists of primary trusted servers and untrusted entities. *Id.* at 12:48-55 (“The identification numbers of trusted agents 120 or trusted servers 200 which have been identified as untrusted are placed on an untrusted list and distributed by the primary trusted servers 210 to the trusted servers 200 and ultimately to the trusted agents 120 in the same fashion as the PTS(PK) list. Merchants which are deemed untrustworthy will have their trusted servers 200 decommissioned by the Trusted Agency and made identifiable to the trusted agents 120.”). The list of untrusted entities is used to identify certificates that have been compromised. *Id.*

E34. Another ongoing security operation of the Trusted Agency includes recertifying trusted agents and servers (certificates expire). Ex. 1020 (Rosen) at 12:16-33.

E35. When a trusted agent's certificate is about to expire, it can contact a trusted server to update its certificate. *Id.* at 15:14-20. The trusted agent will establish a session with the trusted server using the standard certificate exchange process. *Id.* at 15:21-16:59. As I explain in more detail below, this process involves exchanging digital certificates and other standard cryptographic techniques. As part of the process, the server will verify the authenticity of the agent using a handshake procedure involving verification messages (*id.* at 15:46-16:26), and it will check whether the agent is on the untrusted list, (*id.* at 15:30-45). If the server determines the agent cannot be trusted, the session will terminate. *Id.* at 15:38-45.

E36. If the server determines the trusted agent can still be trusted, it will generate a new certificate. The trusted agent first generates a new public private key pair, and transmits it to the server. *Id.* at 16:60-17:7. The trusted server validates the public key, creates a new certificate, and transmits it to the trusted agent. *Id.* at 17:7-22. The trusted agent verifies that the certificate is valid, and if so, it commits the transaction. *Id.* at 17:22-39.

E37. The process I just described also is depicted in the flowcharts in Figures 8A-8C.

4. Communications and Transfers

E38. Rosen shows that content will be exchanged between devices using cryptographically secure channels. Ex. 1020 (Rosen) at 2:13-15, 9:5-15, 17:49-63, 23:12-16, Figs. 12A & 12B. As Rosen explains, each trusted agent can “establish[] a cryptographically secure session” with agents in other transaction devices. Ex. 1020 (Rosen) at 2:13-15.

E39. When initiating a secure channel, trusted agents will exchange digital certificates, and each will validate the other agent’s certificate to ensure the agent can be trusted. *Id.* at 9:5-15, 37:18-38:20. If the validation processes are successful, the trusted agents will negotiate a session key and establish an encrypted channel. *Id.* at 15:21-16:51, 37:18-38:20.

E40. Rosen describes the operation of the secure “Establish Session” protocol in several different exemplary scenarios. It describes the process in the context of obtaining a new digital certificate, as well as in the context of establishing a session with a money module. *E.g.*, Ex. 1020 (Rosen) at 15:14-17:39, 37:19-20; *see also id.* at 9:5-15 (“A Session Manager function 142 sets up and breaks down inter-agent sessions and agent to trusted server sessions. A Security Manager function 144 maintains security information (e.g., a trusted agent

certificate and an untrusted agent list) and establishes secure communication with a counterparty trusted agent (via the host processor 124)"); *id.* at 13:30-34. This is the same process that would be used when initiating a connection between any two trusted agents.

E41. The session initiation protocol involves the exchange of multiple messages between the client and server. Rosen explains that a server will not initiate a session with a client unless the client has a valid digital certificate and the certificate is not on the hot list or bad id list of revoked certificates:

Maintain Security A sends its certificate to Session Manager A which sends it to B (steps 1476-1478). Session Manager B receives the certificate and Maintain Security B (if B is a security server, then this function is performed by the Session Manager) validates the certificate (steps 1480-1484). If the certificate is not valid, then Session Manager B notes the session is terminated . . . If the certificate is valid, then Maintain Security B checks if A is on the bad id list (steps 1494-1496). If A is on the list, then the session is terminated.

Id. at 37:18-38:20. Similarly, a client will not establish a session with a merchant, unless the merchant has a valid digital certificate. *E.g., id.*

E42. In the first message of the session initiation protocol, Rosen shows that the client trusted agent will transmit its digital certificate to the merchant server's trusted agent. Ex. 1020 (Rosen) at 17:61-63, 37:29-35.

E43. The merchant will validate the certificate, and it will verify the customer trusted agent's id number is not on the untrusted list distributed by the Trusted Agency. If the certificate is valid, the merchant trusted agent will create a message containing the merchant's certificate, a first "verification message" (which is a random number), and a first random number that will be used to form the session key. Ex. 1020 (Rosen) at 37:41-49 ("Maintain Security B assembles R(B), B verification message, time and date, and B's certificate in a message []. Public Key B encrypts the message with A's public key and Session Manager B sends the message to A []."). This message is sent to the customer's trusted agent. *Id.*

E44. The customer validates the merchant's certificate, and if valid, it will create a message containing the first verification, a second (newly generated) verification message, a second (newly generated) random number. *Id.* at 37:50-64, 37:64-38:2. This message is sent to the merchant.

E45. The merchant will check to be sure the first verification message is correct. If it is, the merchant will generate a session key from the first and second random numbers and note the start of the session. The merchant will create a message containing the second verification message and send it back to the client. *Id.* at 38:14-20.

E46. The client will check to be sure the second verification message is correct. If it is, the client will generate a session key from the first and second random numbers and note the start of the session. Ex. 1020 (Rosen) at 37:63-65.

E47. As part of the session initiation protocol, Rosen shows that each of the client and the server transmits a “verification message,” which is a random number that is used as part of a handshaking process to prevent replay attacks. By generating a new random number for each communications session, it ensures that old messages (from a different session) cannot be sent back to the device for re-processing because the server will detect that the random number included in the message is incorrect. These numbers can be thought of as a session identifier or a nonce. Rosen explains the verification process:

- Each device creates a “verification message” containing a random number. *Id.* at 15:46-51 (“Random Number Generator B creates a random number R(B) and also a B verification message (step 318). . . The B verification message is a random number used by B to protect against message replay.”), 16:1-4, 37:42-47, 37:62-64.
- The recipient of a verification message will process it and return it to the device that created it. *Id.* at 16:13-18 (“Session Manager A assembles a message containing the A and B verification messages, the date/time information, and R(A) (step 344) . . . and sends the encrypted message to trusted server B[]”), 16:33-36, 37:67-38:1, 38:14-15.
- The device that created the verification message will validate that the recipient processed the message correctly, and if not, will terminate the

session. *Id.* at 16:21-26 (“Security Manager B checks if the B verification message received from A is the same B verification message it previously sent to A (steps 354 -356).

- If it is not the same, then the session is terminated (steps 310-312). If it is the same, then Session Manager B notes the start of the session (step 358).”), 16:54-59, 38:6-8, 38:15-20.

E48. There were several standard variations of this handshaking process, and a person of ordinary skill would have been able to substitute any of those processes into Rosen without changing its function. For example, a person of ordinary skill would understand that where a CTA and MTA establish a session, it is irrelevant to the security of the system whether the CTA or the MTA sends the first message of the Establish Session protocol. The protocol, and the security it provides, are not affected by which agent initiates it. In fact, when Rosen describes the protocol, it generically describes the parties as “A” and “B.”

E49. The process described in Rosen is an example of a 3-way handshake. It would have been a routine task to modify the protocol to use, for example, a 2-way handshake instead. For example, several variations on the handshaking process are described in the Applied Cryptography textbook as well as in Denning. Ex. 1017 (Denning) at 175, 178-79; Ex. 1041 (Applied Cryptography) at 426-28.

E50. In a 2-way handshake, the client device would first request the merchant’s digital certificate. Then the client would create a message containing

its own digital certificate, a nonce, and a session key. Ex. 1041 (Applied Cryptography) at 426-28; Ex. 1037 (X.509) at 23-25. A person of ordinary skill in the art would have considered it to be an obvious design choice to configure the Rosen protocol so that in the first step, the customer trusted agent requested the merchant's digital certificate instead of sending its own certificate to the merchant. Then, in the second step, the customer would transmit a message to the merchant including the customer's certificate, a first verification message, a first random number, and the date and time. The remaining steps of the protocol would be the same.

E51. As part of the session initiation protocol, Rosen shows that trusted agents will negotiate a session key for encrypting subsequent communications. The key is negotiated by exchanging by random numbers that each device uses to generate the key. *See* Ex. 1020 (Rosen) at 15:46-49 ("Random Number Generator B creates a random number R(B) . . . The random number R(B) will eventually be used to form a session key."), 16:1-4, 16:7-9 ("Security Manager A then forms and stores session key (TA/TA) by exclusive ORing random numbers R(A) and R(B) (step 344)."), 16:27-28, 37:42-44, 37:62-65, 38:6-8. Again, this is a standard key negotiation process that was well-known in the art.

E52. The processes I just described also are depicted in the flowcharts in Figures 9A-9E and 38A-38E of Rosen.

5. Merchants and the Transaction Protocol

E53. Rosen explains that a user can connect to a variety of merchants using the transaction device. As Rosen explains:

FIG. 5 shows the general network architecture of the contemplated system for open electronic commerce. Customer transaction device 188 can communicate with a merchant through an gateway network 190 without revealing the owner. **Thus, customer can travel the networks anonymously, paying each in real-time for access. They can search out merchants' electronic space and enter it anonymously, select the item for purchase, and deliver payment in real-time. . . .**

In the preferred embodiment, **the gateways 190 provide CTDs 188 with access to local merchant networks 134 for commerce** and local identification authority networks 192 for acquiring and revalidating credentials (e.g., driver's licenses, credit cards, etc.) **Merchant networks 192 may consist of merchant servers 194 that provide a merchandise catalogue, merchant transactor devices 198 to deliver goods for payment, and merchandise servers 196 which constitute an electronic warehouse.** Merchant networks 192 also preferably have trusted servers 200 for distributing security information.

Id. at 10:6-29 (emphasis added); *see also id.* at Fig. 5. As the passage shows, there are multiple merchant networks, which may have multiple merchant servers.

E54. First, a customer identifies the content he wishes to purchase, and the customer's transaction device sends a message to the merchant's transaction device. Ex. 1020 (Rosen) at 17:51-57, 18:35-38. Each device then passes the process to its trusted agent, which will handle the transaction security.

E55. The customer trusted agent (CTA) then initiates a session with the merchant trusted agent (MTA) using the "Establish Session" protocol I described above. *See id.* at 17:49-63; *see* ¶¶ E41 to E52, *above*. While Rosen shows the customer trusted agent initiating the session, the merchant trusted agent could do so instead. Which trusted agent sends the first message of the Establish Session protocol is irrelevant the system's security.

E56. After the session is established, the MTA sends the customer a merchant "credential ticket." *Id.* at 18:13-16; *see id.* at 18:5-12, 5:4-7. The CTA uses this credential ticket to verify the MTA is an authorized merchant. *See* ¶ E27, *above*.

E57. The MTA then will prepare a new electronic object for delivery to the customer. The MTA encrypts the object with a random key, and then generates a new decryption ticket for the object. *Id.* at 18:67-19:14. The MTA also "digitally signs the encrypted electronic object with [its] private key," and its signature and digital certificate are included in the decryption ticket *Id.* at 19:3-6, Fig. 2. The decryption tickets are generated on the fly in response to customer requests. The

MTA then transmits (using the secure channel already established) the encrypted object and decryption ticket to the CTA. *Id.* at 17:49-60, 18:66-19:14, 23:13-22.

E58. The CTA “verifies the encrypted electronic object’s signature using [the MTA’s] public key . . . , and [i]f the signature is incorrect, then the transaction is aborted.” *Id.* at 19:17-20. The CTA performs additional validations on the object and decryption ticket, (*id.* at 19:20-35). If everything is valid, the CTA stores the decryption ticket in its secure memory. *Id.* at 19:35-39 (it “sends the decryption ticket to Ticket Holder A for storage”). It would have been obvious to configure the CTA to validate the “Usage” field during this process to ensure that the correct rights were received and that the received rights would permit the customer to render the content in the object.

E59. Because the trusted agent stores the decryption ticket (which contains the decryption key) in secure memory, the encrypted electronic object can be stored in the transaction device’s memory because it will be inaccessible to the customer or a transaction application unless the CTA provides access to the decryption ticket. Rosen explains that the CTA will not allow access to the decryption ticket until the transaction is complete. *Id.* at 19:40-43 (“the merchandise is inaccessible to the customer due to its encryption and/or its storage within his trusted agent”), 23:16-21, 4:65-5:3. Storage of the ticket and object are “provisional” until the transaction is committed. *Id.* at 2:17-19, 23:4-6.

E60. Next, the CTA needs transmits a payment credential ticket to the MTA to show it can pay for the electronic object. *Id.* at 5:4-7, 19:46-49, 24:21-29. The MTA will validate the credential and confirm that the payment can be processed. *Id.* at 24:29-49. If so, the MTA will send an acknowledgement message to the CTA. Without a valid payment credential, a customer cannot receive an electronic object, and therefore, it cannot use electronic objects without one.

E61. The trusted agents will then commit the transactions. *Id.* at 22:65-23:9, 24:41-49. The CTA permanently stores the decryption ticket, and passes the electronic object to the transaction device for storage and use. *Id.* at 23:1-9, 23:41-51.

E62. While Rosen describes the purchase transaction where the MTA first sends the content to the CTA and then the CTA sends a payment credential to the MTA, Rosen explains “the order of exchanging electronic merchandize and money may be reversed.” By reversing the exchange, the CTA first sends its payment credential ticket to the MTA, the MTA validates the credential ticket, and if valid, the MTA will transmit the electronic object and decryption ticket to the CTA. *Id.* at 19:13-39, 22:20-67, 23:23-40, 23:52-61.

E63. As I mentioned, the trusted agents do not retain the funds or content until each receives a confirmation signal from the other. This is an example of a

two-phase commit protocol to exchange money and content securely. *Id.* at 18:67-23:8; *see also id.* at 13:36-15:11. The trusted agents provisionally exchange the funds and content, and then each verifies what it has received. If the funds or content are valid, each will send a confirmation message to the other. If both confirm, then they commit the transaction and keep the funds or content. If a trusted agent encounters a problem during this process, it will send an abort signal to the other. If a trusted agent receives an abort signal it removes the content or funds from its memory, and backs out of the transaction.

E64. Rosen summarizes the eight steps of a purchase transaction as follows:

- (1) a secure transaction session is established between the buyer's and seller's money modules, between the buyer's and seller's trusted agents, and between the money module and trusted agent of each transaction device;
- (2) selected electronic merchandise is transferred from the seller's trusted agent to the buyer's trusted agent (where it is retained provisionally)--in the event that the electronic merchandise includes an electronic object, the electronic object is encrypted so that it may be stored outside of the trusted agent;
- (3) after verifying the correctness of the transferred electronic merchandise, the buyer's trusted agent instructs its money module to pay a certain amount of electronic money to the seller's money module;

- (4) the buyer's money module informs the seller's money module of the amount of electronic money to be paid to it and the seller's money module checks with its trusted agent to verify that this is the correct price of the merchandise;
- (5) if the amount is correct, the seller's money module sends an acknowledgement to the buyer's money module;
- (6) the buyer's money module transfers the electronic money to the seller's money module (the seller's MM provisionally retains the note(s) and the buyer's MM provisionally decrements the value of the note(s) in the transferred amount);
- (7) both the buyer's and seller's money modules commit (the seller MM's retention of the note(s) is no longer provisional and the buyer's MM retains the new value(s) of the note(s)) and, in so doing, send "payment successful" messages to their respective trusted agents;
- (8) finally, both the buyer's and seller's trusted agents commit (the seller's trusted agent records the sale and the customer trusted agent's retention of the merchandise is no longer provisional), so that the buyer can now use his/her electronic merchandise and the seller has his/her electronic money.

Ex. 1020 (Rosen) at 23:9-51

E65. The purchase process also is depicted in the flowcharts in Figures 12A & 12B, 15A & 15B.

6. Acquiring Credentials

E66. Rosen also describes a protocol for acquiring a credential ticket. Ex. 1020 at 26:44-27:30, Fig. 26.

E67. Both CTAs and MTAs can acquire credentials from a special trusted agent, an Identification Authority trusted agent (“ATA”), using the same process. *See id.* at 26:44-49, 26:49-53 (“The credentials can be . . . acquired remotely if the trusted agent 120 already contains credentials for proof of identity. . . .”). While Rosen shows ATAs are being part of a separate network than merchants (*see id.* at Fig. 5, 10:5-37), a particular merchant could host its own ATAs that could be used to identify customers or provide custom payment options.

E68. For example, for a CTA to obtain a credential ticket, the CTD sends a request to an Identification Authority’s transaction device. Ex. 1020 (Rosen) at 26:44-27:30. Each transaction device will pass the process to its trusted agent for security purposes. The ATA will evaluate the request. For example, if the requester is asking for a payment credential from a bank, the ATA will check the requester’s identity and make sure the requester owns an account. *Id.* at 26:51-52, 26:57-59, 27:53-59. Or, if the requester is asking for a merchant credential, the ATA will check the requester’s identity and make sure the requester is in fact a merchant. *Id.* Rosen also shows that in some circumstances, acquiring the credential ticket cannot be done remotely, and if that is the case, the ATA will

deny the request and inform the request he or she must obtain the credential in person. *Id.* at Fig. 26.

E69. If the ATA determines that request can be granted, it will use the “Establish Session” protocol to initiate a session with the requesting trusted agent. , create a new credential ticket, and then transmit the ticket to the CTA. *Id.* at 27:5-14, 27:62-65.

B. Overview of Hartrick

E70. Hartrick describes a technique for controlling the use and distribution of “softcopy” (*i.e.*, digital) books. Ex. 1021 (Hartrick) at 1:3-8, 5:19-38.

E71. Each softcopy book is a “structured document” that has been formatted using the Standard General Markup Language (SGML). SGML is a markup language that evolved into the eXtensible Markup Language (XML) that is in common use today. Markup languages like SGML use “tags” to describe each “element” or component of a document. *Id.* at 4:25-36, 5:3-8, 5:42-47. Tags typically are denote with “<” and “>”, although Hartrick uses “[“ and “]” to denote tags. *See* Ex. 1062 (HTMLv2.0) at 2 (“SGML is a system for defining structured document types and markup languages to represent instances of those document types[SGML].”), 10-12 (“The SGML declaration determines the lexicon of the grammar. It specifies the document character set, which determines a character repertoire that contains all characters that occur in all text entities in the document,

and the code positions associated with those characters. . . The SGML declaration also specifies the syntax-reference character set of the document, and a few other parameters that bind the abstract syntax of SGML to a concrete syntax. This concrete syntax determines how the sequence of characters of the document is mapped to a sequence of terminals in the grammar of the prologue.”).

E72. Each user has a softcopy book reading program installed on its computer that can display or render the book in a form viewable by a user. Ex. 1021 (Hartrick) at 6:1-9. The reading program can interpret the SGML tags to determine how the book should be formatted and displayed to a user, amongst other things. *Id.* at 1:12-16, 2:21-28.

E73. Hartrick shows the softcopy book reading program is able to interpret a library of different tags, and use the tags to format the structured the document for display to a user. For example, Hartrick shows text formatting tags, such as the “[bk]” tag to denote a book’s title (“[bk] Book Title [/bk]”) and the “[p]” tag denotes a paragraph of text (“[p] Paragraph of text...[/p]”). *Id.* at 1:28-39, 4:38-42, 9:41-49, 11:37, 12:45.

1. Specialized SGML Tags

E74. Hartrick explains that the library of SGML tags used in its system includes two types of specialized tags. The first is security tags, which are used to specify how the book can be used and if it can be distributed by a user. The second

is royalty tags, which are used to specify that a user can reproduce the book, but only if the user pays a fee. *Id.* at 1:3-8, 3:12-22, 3:28-32, 5:48-52.

E75. Hartrick shows how these specialized SGML tags can be incorporated into each book, and that this allows authors and distributors to better manage and collect royalties from end-users. *E.g.*, Ex. 1021 (Hartrick) at 3:53-58, 10:1-14.

E76. To be able to view books in the Hartrick system, each end-user's computer also needs a special program for interpreting the security and royalty tags. Hartrick explains that each end-user's computer has a royalty payment program running on it, and that the program will intercept requests from the user to store, copy, print, and transmit a book or portions thereof. *Id.* at 6:9-22, Fig. 1.

E77. When the Hartrick royalty payment program intercepts requests to print/copy/transmit a book, it interprets the tags associated with the softcopy book to determine whether it contains tags restricting the user from taking that action or whether the book contains tags allowing the user to take the action but only for a fee. *E.g.*, *id.* at 16:10-21.

E78. Hartrick explains that the use of specialized SGML tags allows authors and distributors to exercise some granularity of control over end-users' ability to reproduce the book. Hartrick describes several types of reproduction that the technique can control: "printing pages of a structured document," (*id.* at 5:24-26,), "writing [] a structured document into bulk storage," (*id.* at 5:30-32), and

transmitting via “telecommunication [] soft copies of a structured document,” (*id.* at 5:35-37).

E79. The Hartrick system allows the security tags and royalty tags to be integrated “within the structured document text of the book” or the tags can be distributed “in a royalty payment information file which accompanies the book.” *Id.* at 5:48-52.

E80. Hartrick describes “security labels” as “special structured document tag[s]” that are interpreted by the royalty payment program to prohibit certain operations on the book. *Id.* at 3:28-32, 3:48-4:9. The security labels, which actually are “security tags,” can be used to specify how the softcopy book can be used or distributed. *Id.*

E81. While Hartrick describes several examples of security tags, which illustrate the role that security tags play in the system.

E82. Hartrick describes a “Do Not Copy” tag, which the royalty payment program interprets as allowing a user to display or view the book on a screen, but as inhibit a user from printing, copying, or transmitting the book. *Id.* If a user requests, for example, to print a book that contains this tag, then any requested “printing operation[s] will be aborted in response to the [tag’s] presence.” *Id.* at 3:53-57. The same goes for requests to transmit or copy the book. *Id.* at 3:53-4:9.

E83. Hartrick describes a “Do Not Distribute” tag. Although Hartrick does not explicitly state what the tag does, a person of ordinary skill would have understood that the “Do Not Distribute” tag would be used to prevent the document from being transmitted, but that it would permit the book to be both displayed and printed. *Id.* at 3:28-32, 4:4-9.

E84. Hartrick describes a “copyright” tag that will cause every page of the displayed document to contain a copyright notice. *Id.* at 3:33-40, 50-53.

E85. Hartrick describes several examples of possible security tags. However, many more different types of security tags are possible. A person of ordinary skill would understand that Hartrick was not trying to describe a specific limited set of security tags, but rather, that Hartrick was describing a technique where specialized SGML security tags are used to control how digital content is reproduced on an end-user’s computer.

E86. Although Hartrick describes security tags as restricting a user’s ability to take certain action, a person of ordinary skill in the art implementing a system based on Hartrick would have found it obvious to use tags grant users rights instead. For example, instead of using a “Do Not Copy” tag to prevent a user from doing anything but viewing a document, one could use a “Display” tag to indicate the user had the right to display the document. This would make it simple to add additional rights such a “Print” tag or a “Transmit” tag. SGML is extensible, and it

was routine to create a different library of custom tags for use in particular systems.

E87. Hartrick describes another type of tag that it calls royalty tags.

Royalty tags allow an author or publisher to specify different fees that must be paid to reproduce the book or any part of the book (*e.g.*, a different fee for each chapter). *E.g.*, Ex. 1021 (Hartrick) at 5:42-47, 10:1-14, 10:41-54, 11:46-12:36.

E88. Each royalty tag is accompanied by an “amount” tag, which specifies the fee for taking an action. Hartrick shows an example of these tags in a document:

The book title portion of the second example document 25 of Fig. 4, has the following elements:

"[bk] Book Title [/bk]" 300,

"[ed] Second Edition [/ed]" 302,

"[cpr] (C) ABC Co 1990 [/cpr]" 304,

"[royalty] Book Reproduction Fee [/royalty]" 306,

"[amount] \$20.00 [/amount]" 308,

"[phone] 1-800-123-1234 [/phone]" 310,

"[public key] 13A723F9...6 [/public key]" 312, and

"[validation] The Book Repro Fee Is Paid [/validation]" 314.

Ex. 1021 (Hartrick) at 11:35-45. And:

The first chapter portion of the second example document 25 of Fig. 4, has the following elements:

"[h1] First Chapter Heading [/h1]" 316,
"[royalty] Chapter Reproduction Fee [/royalty]" 318,
"[amount] \$ 1.00 [/amount]" 320,
"[validation] First Chapter Fee Paid [/validation]" 322,
"[p] Paragraph of text [/p]" 324,
"[h2] First topic heading [/h2]" 326,
"[p] Paragraph of text [/p]" 328,
"[p] Paragraph of text [/p]" 330,
"[h2] Second topic heading [/h2]" 332,
"[p] Paragraph of text [/p]" 334,
"[h3] First subtopic heading [/h3]" 336,
"[p] Paragraph of text [/p]" 338, and
"[p] Paragraph of text [/p]" 340,

Ex. 1021 (Hartrick) at 12:37-53.

E89. Unlike the security tags, which either grant or restrict a user from reproducing a softcopy book in a certain fashion, the royalty tags grant a conditional right to reproduce the book. The condition is specified using the “amount” tag that follows the royalty tag. If the user satisfies the condition by paying the fee, the action specified by the royalty tag can be performed.

E90. Hartrick depicts several examples where a general “royalty” tag governs any type of “reproduction” of a book (*e.g.*, printing, copying, *and* transmitting). *E.g.*, Ex. 1021 (Hartrick) at 11:35-45. However, just as with the security tags, a person of ordinary skill would understand that Hartrick was not describing a particular limited set of SGML tags to be used to protect books. Instead, Hartrick was describing a technique for using specialized SGML tags to grant conditional rights to a digital content file.

E91. Hartrick explains how security and royalty tags can be used to govern a range of different actions that could be taken by users. Throughout the specification and claims (which were published with the application), Hartrick distinguishes between several different actions users can take with a softcopy book, such as printing the book, making a copy of it, and transmitting it over a network other computers. *E.g.*, *id.* at 5:19-38.

E92. Hartrick also shows that a publisher can separately provide authorization for taking the different actions. In the published claims, Hartrick shows that when a user request to reproduce a book, the book royalty payment program sends a request to the publisher including the type of operation (*e.g.*, copy, print, transmit), and that the publisher returns to the user a digitally signed authorization message that specifically authorizes the requested operation. *E.g.*, *id.*

at 21:56-22:3, 22:7-10 (print); 22:34-38, 22:42-45 (copy); 23:13-14, 23:21-24 (transmit).

E93. Although Hartrick provides several specific examples of types of royalty tags, a person of ordinary skill in the art would have understood Hartrick to teach that different royalty tags could be used to specify fees for the different operations Hartrick is designed to control.

E94. When implementing Hartrick, it would have been obvious that distinct tags could be used for each of printing, copying, and transmitting a book (*e.g.*, “[print-royalty]” and “[transmit-royalty]”). *See, e.g.*, Ex. 1021 (Hartrick) at 5:19-38, 6:9-22, 10:7-18. Using distinct tags offers greater granularity of control over the use and reproduction of a book, and would allow for different fees for different action, such as one fee for printing and another for making a digital copy.

E95. Hartrick shows that the specialized SGML tags can apply to a book as a whole, or they can apply to particular chapters or paragraphs. Ex. 1021 (Hartrick) at Fig. 4, 11:35-45, 12:37-53. Hartrick also shows that several books can be distributed together, in a single file. Ex. 1021 (Hartrick) at 9:29-31.

E96. Hartrick shows the tags can be embedded in the document, or they can be in a separate file that is attached to the document. *See* Ex. 1021 (Hartrick) at Fig. 10. Where tags are distributed in a separate file, they could be associated with different parts of the document using the “coordinate” technique described in

Hartrick. *See id.* at Fig. 4 & 7. This would be a simple mechanism for making the security and royalty tags easily accessible for verification purposes, yet still provide the granularity of control described in Hartrick.

E97. One of the benefits of using a language like SGML, is that it is easy to change the available tags and syntax between different implementations. A person of ordinary skill would have found it to be a routine task to adjust the available tags to fit the details of any particular project.

E98. For example, a person of ordinary skill would find it obvious to use “display” tags or “print” tags to govern use of a document. If the tags were implemented in a system for distributing software, it would have been obvious to use tags such as “execute” or “transfer.” It also would be obvious to specify different kinds of conditions. For example, there could be an “expiration date” tag, a “max use” tag, or a “user id” tag that could be used to restrict when or who used the content. These are all routine and obvious variations on the concept described in Hartrick.

2. Royalty Payment Program

E99. A royalty payment program on the user’s computer intercepts the user’s requests and enforces the specialized SGML tags. *Id.* at 6:9-22, Fig. 1. Hartrick shows that the royalty payment program will not permit the user to perform the requested operation unless the security tags permit the operation (*id.* at

3:30-32, 3:53-4:9), and the user has paid any fees specified by the royalty tags (*id.* at 15:34-40, 16:10-21, 20:1-21). For example:

If the user enters a command to copy the book onto a writable storage medium such as a magnetic disk or to print a hardcopy of the book with a printer or to transmit a copy of the book over a modem, the royalty payment program intercepts the copying command and suspends the copying operations.

Id. at 6:9-19; *see id.* at 3:48-4:9, 10:7-18, 15:23-33.

E100. If the royalty payment program intercepts a request, and the user has not yet paid the fee, “the user must select the option of paying a royalty to the publisher before the royalty payment program permits a copy of the book to be made.” *Id.* at 6:19-22. The royalty payment program will then send a request to the publisher for authorization. *Id.* at 15:34-40, 15:53-16:9.

E101. Hartrick explains that the publisher’s server runs a royalty billing program that processes authorization requests from end-users. *Id.* at Figs. 2 & 9, 16:10-33, 6:34-7:10.

E102. The royalty billing program is configured to determine whether the user can pay for the requested operation, and if so, it returns an “authorization message.” Only after the user obtains authorization will the royalty payment program permit the requested action to proceed. *Id.* at 16:18-21, 20:5-17. The

user's royalty payment program will verify the message and then permit the operation to proceed. *Id.* at 16:10-21.

E103. Hartrick shows the authorization message can be a digitally signed message that contains data indicating that the requested operation is allowed. *Id.* at 16:26-32, 17:18-26, 18:46-19:16. The authorization message contains data specifying which operation (*e.g.*, print or transmit) the user has permission to take. *Id.* at 22:7-10, 42-45; 23:21-24. It also can contain data specifying a maximum number of pages for which the action is granted and a page range. *Id.* at 16:34-37, 20:5-8, 20:31-35).

C. A Person of Ordinary Skill Would Have Integrated Hartrick's SGML Tag Technique in the Rosen System

E104. Rosen shows that the decryption ticket contains a "Usage" field that is used to specify "restrictions on usage of the electronic object." Ex. 1020 (Rosen) at 6:12-13. However, Rosen does not provide any details about how these restrictions would be implemented and enforced.

E105. To utilize the "Usage" field, a person of ordinary skill in the art would have turned to known techniques for specifying and enforcing usage conditions such as those described in Hartrick. Using specialized SGML tags as described in Hartrick would be an obvious way to implement this functionality into the Rosen system. For example, the SGML tags can be easily adapted for different implementations and offer a high level of granularity of control. *See* ¶ 71, *above*.

E106. A person of ordinary skill also would recognize that implementing the Hartrick softcopy book system using the trusted devices described in Rosen improve the security of the Hartrick scheme. Use of tamper-resistant modules was a well-known method of increasing system security. *See ¶¶ A62-A90, above.*

E107. Integrating the Hartrick scheme into an implementation of the Rosen system would be a routine engineering task for a person of ordinary skill, requiring only minor modifications to the two systems.

1. Storing SGML tags in the “Usage” field

E108. A person of ordinary skill would have found it logical to store security and royalty markup statements, such as those described in Hartrick, in the “Usage” field, and would have modified the MTA to include the specialized SGML tags when creating each decryption ticket. The security and royalty tags would allow an MTA to specify in each ticket the different manners in which the customer could use the electronic object.

E109. For example, the MTA could use a “Do Not Copy” tag (which prevents a user from copying, printing, or transmitting) to specify that a user could only electronically display an electronic object, or a “Do Not Distribute” tag to specify that a user could both display and print the electronic object.

E110. The MTA also could use different types of royalty tags to specify additional rights a user could exercise for a fee, such a “print-royalty” tag

specifying a fee for printing (*e.g.*, a nominal fee since the user already purchased the electronic version), and a “copy-royalty” tag specifying a different fee for making a copy (*e.g.*, full price). *See* Ex. 1021 (Hartrick) at 3:28-32, 5:23-38, 6:9-22, 9:11-23.

E111. A person of ordinary skill integrating the schemes would also have understood that the security tags could be implemented to grant affirmative rights—*e.g.*, a “Display” tag (instead of “Do Not Copy”), or a “Print” tag and a “Display” tag (instead of “Do Not Distribute”). This could be done by having separate “Display” and “Print” tags, or by using a royalty tag with no fee, for example “[display-royalty] \$0.00 [/display-royalty]”.

E112. Similar to decryption tickets in Rosen, Hartrick shows the royalty and security tags can be stored in a separate royalty information file that is distributed with the content. *See, e.g.*, Ex. 1021 (Hartrick) at 5:48-52. As I explained above, the tags could be associated with different parts of the content using the “coordinate” technique described in Hartrick. *See* ¶¶ E95-E98, *above*. Integrating the tags into the Rosen decryption ticket is a logical extension of both Rosen and Hartrick.

E113. In the combined system, the Hartrick book viewing program could be distributed as one of Rosen’s “transaction applications” for viewing content. It could be distributed in an electronic object and be accompanied by a decryption

ticket. This would allow all the features of the Hartrick system to be implemented on top of Rosen. A trusted agent would install the program only after validating the digital signature and certificate. *See* ¶ E58, *above*.

E114. For example, this would allow Hartrick's softcopy books to be distributed using the Rosen system. The specialized SGML tags would be stored in the "Usage" field of the decryption ticket, and could be associated with each chapter of the book using the coordinate technique. *See* ¶¶ E95-E98, 112, *above*.

In addition, I note that, although Hartrick describes its technique in the context of softcopy books, a person of ordinary skill could apply it to any type of electronic object distributed via Rosen's system, and modify the tags to reflect actions associated with other types of content. *See* Ex. 1021 (Hartrick) at 9:29-31 ("elements can also include graphics as well as text"); *see* ¶ 71, *above*..

2. Modifying the Trusted Agents

E115. A person of ordinary skill could have integrated the royalty billing functionality into Rosen's MTA, allowing the merchant to act as an authorization server for processing requests for additional rights, such as a request to print a paper copy of a book. *See* ¶¶ E99-E103, *above*. In this configuration, when a customer requested to take an action requiring payment of an additional fee (*e.g.*, to print an object where there is "print-royalty" tag), its CTA would send a request to the MTA. *See* Ex. 1021 (Hartrick) at 15:53-16:9, Fig. 8. The MTA would determine

if the request was valid and if the user could pay the royalty fee. *See* Ex. 1021 (Hartrick) at 16:10-17, Fig. 9. If valid, the MTA could return a digitally signed authorization message as described in Hartrick (*id.* at 17:18-26, 18:32-19:16), which could include data about which operation was permitted (*e.g.*, copy or print) and on which pages (*id.* at 16:34-17:6, Fig. 8A).

E116. Rosen shows the MTAs have analogous functionality, as they are configured to provide credential tickets and decryption tickets in response to requests from customers. Configuring the merchants to sell additional rights to the same content by selling an additional type of “ticket” (*e.g.*, the Hartrick authorization message) would be a simple and obvious extension of the Rosen system.

E117. A person of ordinary skill would have integrated the rights checking functionality of Hartrick’s royalty payment program into the CTA’s protocols. Integrating security functionality into a tamper-resistant component was a conventional, well-known process. *See* ¶¶ A62-A90, *above*.

E118. In this configuration, whenever a user requests a transaction application to render, copy, print, or transmit an electronic object, the CTA intercepts the request and checks the requested operation against the statements in the “Usage” field to determine whether the operation is authorized. *See* ¶¶ E99-E100, *above*. If the operation is not authorized, the CTA can either deny the

request or initiate a connection with the MTA to pay the royalty and obtain authorization. If the action is permitted or authorization is obtained, the CTA will permit the action and allow access to the decrypted content. *See* ¶¶ E100-E103, *above*.

E119. In addition, a person of ordinary skill would have recognized the advantages of integrating the rights checking protocol into a MTA to allow authors to specify enforceable usage restrictions when providing content to a merchant.

E120. Rosen shows that merchants store content in cleartext form and a MTA will generate decryption tickets on-the-fly in response to user requests. Ex. 1020 (Rosen) at 18:66-19:12; *see* ¶¶ E57, E116, *above*. Hartrick, on the other hand, explains that the author specifies usage limitations (*e.g.*, royalty fees) when depositing a book with the publisher. *See* Ex. 1021 (Hartrick) at 9:11-14.

E121. It would have been obvious that in the combined system, when an author deposits content with the merchant, the merchant encrypts the content and the MTA creates and maintains a decryption ticket in which the “Usage” field contains the author’s usage restrictions and royalty fees (as tags described in Hartrick). By encrypting the content, the author could be certain its content was secure on the merchant servers, and by specifying usage information, the author could enable the MTA to enforce the rights and conditions when selling the content. Again, this is simply incorporating conventional techniques.

3. Tuning the Cryptographic Protocols

E122. A person of ordinary skill would have adjusted Rosen's cryptographic protocols to accommodate the particular implementation.

E123. For example, Rosen shows that during a purchase transaction the CTA will initiate a session by sending the MTA its digital certificate. This configuration is one of several standard variations of the certificate exchange process; a person of ordinary skill could easily have altered the process so that the MTA initiated the session by sending the CTA its digital certificate and integrated the modification into Rosen without any change in function. *See, e.g.*, Ex. 1041 (Applied Crypto) at 426-428 (when exchanging digital certificates, the parties can use a “one-way, two-way, or three-way authentication protocol”); Ex. 1017 (Denning) at 178-79; Ex. 1047 (X.509) at 23-25.

D. Implementing Rosen to Use the Hartrick Scheme Would Create a System that Uses “Usage Rights” and “Repositories”

1. The Rosen / Hartrick System Would Use “Usage Rights”

E124. As I explained, a person of ordinary skill would have specified different rights and restrictions in the “Usage” field using statements in a markup language such as by using the specialized SGML tags described in Hartrick. *See* ¶¶ E78-E98, E108-E114, *above*. In a combined Rosen/Hartrick system, the “Usage” field from Rosen would contain statements in a language that define how the electronic object can be used or distributed.

E125. Each of Hartrick's specialized SGML tags specify different types of permissible actions. For example, the "Do Not Copy" tag grants the user the right to display and the "Do Not Distribute" tag grants the right to display and print. *See ¶¶ E78-E85, above.*

E126. A person of ordinary skill would have adjusted the SGML tags to fit the particulars of a project, and would have implemented these tags to grant affirmative rights instead of as restricting rights. For example, by using a "Display" tag, a "Print" tag, and a "Transmit" tag. This would make it easier to grant users additional rights. *See ¶¶ E85-E98, above.* This also likely would have improved the system's security – it was well-known security principle that security is improved when a user has no rights unless explicitly granted, as opposed to having all rights unless specifically restricted.

E127. As another obvious variation, a person of ordinary skill also could have implemented these rights by using zero-fee royalty tags such as a "display-royalty" tag or a "transmit-royalty" tag that specified a zero-dollar fee. This might be done, for example, to ensure backward compatibility with older versions of the royalty payment program. *See ¶¶ E93-E98, above.*

E128. In the combined Rosen/Hartrick system, the MTA would write the tags in the "Usage" field of the decryption tickets to specify whether a user can copy, print, store, or transmit digital content. This would cause the "Usage" field

to contain data that indicate specific *actions* or “manners of using or distributing” the content. *See* Ex. 1001 (859) at 18:35-19:52 (Possible rights include “play,” “print,” “copy,” and “transfer”).

E129. Hartrick explains that each royalty tag can specify a royalty payment amount that must be made to exercise the right. For example, if a user wants to print a copy of a book, an “[amount]” tag specifies the fee that must be paid. Ex. 1021 (Hartrick) at 12:3-12 (“The associated elements to royalty element [] are ‘[amount] \$20.00 [/amount]’ . . .”).

E130. Payment of the royalty amount is a “condition” on the exercise of the right in the Stefik scheme. *See* Ex. 1001 (859) at 18:21-25 (“These conditions are copy count, control, time, access and *fee conditions*.” (emphasis added)). In Rosen/Hartrick system, the “Usage” field data would specify *conditions* on a specified manner of use.

E131. Rosen shows that each decryption ticket is permanently attached to each electronic object once purchased. Ex. 1020 (Rosen) at 5:59-6:13, 4:54-55 (“A decryption ticket is always associated with a particular encrypted electronic object.”), 4:60-63 (“An encrypted electronic object can be decrypted by unique information in its associated decryption ticket.”), 4:55-5:3.

E132. Because the markup statements (*i.e.*, the “rights” and “conditions”) would be stored in the “Usage” field of the decryption ticket, they are also permanently attached to the electronic object.

E133. A person of ordinary skill would have modified Rosen to incorporate the Hartrick tagging scheme to allow author to establish the usage restrictions and conditions when depositing digital content with a merchant. The merchant would encrypt the content it receives from the author and then create a decryption ticket that specifies those conditions in the “Usage” field. *See ¶¶ E119-E121, above.*

E134. The “usage rights” would be permanently attached to the electronic object when the object is deposited into the system and remain with the object as it is distributed. *See ¶¶ E119-E121, above.*

E135. In a Rosen/Hartrick system, the trusted agents could be modified to incorporate the royalty payment program functionality to enforce the data in the “Usage” field. *See ¶¶ E117-E121, above.* As I explain below, each transaction device containing a trusted agent is a repository, and therefore, the “usage rights” are enforced by a “repository.”

2. A Rosen / Hartrick System Would Use “Repositories”

E136. As I explained above (¶ B88), the Stefik patents define a “repository” as “a trusted system” that “maintains physical, communications and behavioral

integrity,” and it enforces usage rights. Rosen shows that each transaction device is a “repository” within the meaning of the claims.

E137. Rosen explains that each transaction device contains a trusted agent is a tamper-resistant module, and each can be identified by a digital certificate. Thus, each transaction device is a trusted system. *See ¶¶ E28-E37, above.*

E138. Rosen explains that each transaction device contains a trusted agent which provides physical security and ensures a secure protocol is used to exchange, store, and render content. *See ¶¶ E1-E15, E22-E26, above.* Decryption tickets are stored in the trusted agent on the transaction device. *See ¶¶ E8-E9, E13-E19, E58-E59, above.* Electronic objects are stored in encrypted form, and the content in each object can only be accessed if the trusted agent possess the associated decryption ticket. Therefore, the transaction device has physical integrity.

E139. Rosen explains that trusted agents communicate using a secure protocol that involves exchanging digital certificates, encryption, and nonces. Thus, each transaction device has communications integrity. *See ¶¶ E38-E52, above.*

E140. Rosen shows that each decryption ticket contains a digital certificate that is used to validate the digital signature associated with the electronic object. The digital certificate and digital signature are used to authenticate the source of

the content and the content's integrity. If the integrity of an object cannot be validated, the trusted agent will delete the object. Thus, each transaction device has behavioral integrity. *See* ¶¶ E23-E25, *above*. For example, if a user downloaded the Hartrick softcopy book reading program as a transaction application, the trusted agent would install the program only after validating the digital signature and certificate. *See* ¶ E58, *above*.

E141. Rosen explains that a decryption ticket contains a "Usage" field that contains "restrictions on usage of the electronic object." Ex. 1020 (Rosen) at 6:12-13, Fig. 2. As explained above, this field contains statements in a usage rights language. *See* ¶¶ E8, E104-E114, E124-E135, *above*.

E142. A person of ordinary skill would have configured the trusted agents to enforce compliance with the data in the "Usage" field of the decryption ticket. Because the trusted agent stores the decryption key necessary to access the content, it can intercept requests to execute, access, print, or copy the content in an encrypted electronic object. The trusted agent would not permit a transaction application to access the decryption ticket unless the statements in the "Usage" field are satisfied. *See* ¶¶ E117-E119, *above*.

E143. In a merchant transaction device, the trusted agent would enforce the "Usage" conditions when generating a new decryption ticket. The usage restrictions might specify the types of rights that could be sold or a minimum price.

If the conditions specified by the author were not met (*e.g.*, the merchant tried to sell the content for too low a price or tried to sell a right it was not authorized to sell), the MTA would not generate a newly encrypted electronic object or a new decryption ticket. *See ¶¶ E115-E121, above.*

E144. Again, Rosen explains that because the trusted agent is physically secure, the protocols are secure and cannot be modified. *See ¶¶ E1-E10, above.*

E145. The usage information in a digital ticket are “usage rights” within the meaning of the claims.

E. Rosen and Hartrick Would Have Rendered Stefik Patents Obvious

1. The Independent Claims of the '859 Patent

E146. The independent claims the '859 patent describe a “rendering system” that can access content over a network and render content in accordance with usage rights attached to that content. A “rendering device” is coupled to a “distributed repository,” and the distributed repository can request content from another repository and it enforces usage rights when content is rendered by the rendering device.

E147. Rosen discloses the processes described in the '859 independent claims. As explained above, Rosen shows a system for distributing and rendering electronic objects (“content”). Ex. 1020 (Rosen) at 1:5-11, 4:55-57. Electronic objects are exchanged using transaction devices (“rendering system”), which are

computers that contain a trusted agent, a processor, and transaction applications for rendering content. Digital content is stored in electronic objects, and each object is associated with a decryption ticket that contains a usage field for storing usage rights information. As I explained above (§§ E115-E121), it would have been obvious to configure the trusted agent in each transaction device to enforce the usage rights information in the usage field (“render[s] content in accordance with usage rights”).

E148. Rosen shows that the transaction devices are “distributed repositories” that have both a “requester” and “server mode of operation.” Rosen shows a customer transaction device requesting content (“requester mode of operation”) from a merchant transaction device. Rosen shows there can be multiple merchants and a distributed merchant network. As I explained above (§§ E115-E121), it would have been obvious to configure the trusted agent in each transaction devices to enforce the usage rights information in the usage field (“server mode of operation”) when a user requested to use content.

a. Claim 1 of the '859 Patent

E149. The processes I described above disclose all the limitation of claim 1 of the '859 patent. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'859 Patent: Claim 1</u>	<u>Relevant Concepts and Paragraphs</u>
------------------------------------	--

<p>1. A rendering system adapted for use in a distributed system for managing use of content, said rendering system being operative to rendering content in accordance with usage rights associated with the content, said rendering system comprising:</p>	<p><u>“distributed system for managing use of content”</u></p> <p><u>Rosen</u> shows a network of merchants and other authorities for selling content to end users. ¶¶ E1-E4, E28-E37, E53-E69, E115-E116.</p> <p><u>“rendering system”</u></p> <p>Customer transaction device. ¶¶ E11-E20, E53, E117-E121</p> <p><u>“rendering content in accordance with usage rights”</u></p> <p>It would be obvious to store specialized SGML tags in the “Usage” field of the decryption ticket to specify whether content can be displayed, printed, copied, or transferred. ¶¶ E6-E10, E74-E98, E104-E114.</p> <p>It would have been obvious to configure the CTA in customer transaction device to enforce those tags. ¶¶ E21-E27, E99-E103, E117-118.</p>
<p>[1.a] a rendering device configured to render the content; and</p>	<p><u>“rendering device”</u></p> <p>Transaction devices (<i>e.g.</i>, computers) have applications for rendering content. ¶¶ E11-E20.</p>
<p>[1.b] a distributed repository coupled to said rendering device and including a requester mode of operation and server mode of operation,</p>	<p><u>“distributed repository”</u></p> <p>Transaction devices contain trusted agents that are physically secure and that execute secure protocols for acquiring, storing, and rendering content. ¶¶ E2-E10; E24-E37, E136-</p>

	<p>E145</p> <p><u>“mode[s] of operation”</u></p> <p><i>See [1.c] & [1.d] below</i></p>
<p>[1.c] wherein the server mode of operation is operative to enforce usage rights associated with the content and permit the rendering device to render the content in accordance with a manner of use specified by the usage rights,</p>	<p>It would have been obvious to configure the CTA in a customer transaction device to enforce specialized SGML tags stored in the “Usage” field of the decryption ticket. ¶¶ E99-E103, E115-E121</p> <p>It would have been obvious to have authors specify usage information when depositing content with merchants, and to configure the MTA in a merchant transaction device to enforce specialized SGML tags stored in the “Usage” field of the decryption ticket. ¶¶ E99-E103, E115-E121</p>
<p>[1.d] the requester mode of operation is operative to request access to content from another distributed repository, and</p>	<p>Customer transaction devices can request content from a merchant. ¶¶ E21-E25, E53-E65, E66-E69, E99-E100</p> <p>Merchant transaction devices can request content from an Identification authority. ¶¶ E66-E69.</p>
<p>[1.e] said distributed repository is operative to receive a request to render the content and permit the content to be rendered only if a manner of use specified in the request corresponds to a manner of use specified in the usage rights.</p>	<p>A customer uses a transaction application to send a request to the CTAs on the customer transaction device asking to access the decryption ticket and render content. ¶¶ E15-E20, E59, E99-E103, E115-E121, E142</p> <p>It would have been obvious to configure MTAs to enforce the “Usage” field of the decryption ticket when selling content. ¶¶ E119-E121, E143.</p>

b. Claims 29 and 58 of the '859 Patent

E150. Independent claims 29 and 58 of the '859 patent are identical to claim 1, except claim 29 specifies a method and claim 58 specifies software. Rosen discloses systems, methods, and software for implementing the techniques described in the article. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'859 Patent: Claim 29</u>	<u>'859 Patent: Claim 58</u>	<u>Relevant Concepts and Paragraphs</u>
29. A rendering method adapted for use in a distributed system for managing use of content, and operative to render content in accordance with usage rights associated with the content, said method comprising:	58. A computer readable medium including one or more computer readable instructions embedded therein for use in a distributed system for managing use of content, and operative to render content in accordance with usage rights associated with the content, said computer readable instructions configured to cause one or more computer processors to perform the steps of:	<i>See Claim 1.</i>
[29.a] configuring a rendering device to render the content;	[58.a] configuring a rendering device to render the content;	<i>See Claim 1.</i>
[29.b] configuring a distributed repository coupled to said rendering device to include a requester	[58.b] configuring a distributed repository coupled to said rendering device to include a requester	<i>See Claim 1.</i>

mode of operation and server mode of operation;	mode of operation and server mode of operation;	
[29.c] enforcing usage rights associated with the content and permitting the rendering device to render the content in accordance with a manner of use specified by the usage rights, when in the server mode of operation;	[58.c] enforcing usage rights associated with the content and permitting the rendering device to render the content in accordance with a manner of use specified by the usage rights, when in the server mode of operation;	<i>See Claim 1.</i>
[29.d] requesting access to content from another distributed repository, when in the requester mode of operation; and	[58.d] requesting access to content from another distributed repository, when in the requester mode of operation; and	<i>See Claim 1.</i>
[29.e] receiving by said distributed repository a request to render the content and permitting the content to be rendered only if a manner of use specified in the request corresponds to a manner of use specified in the usage rights.	[58.e] receiving by said distributed repository a request to render the content and permitting the content to be rendered only if a manner of use specified in the request corresponds to a manner of use specified in the usage rights.	<i>See Claim 1.</i>

E151. It is my opinion that Rosen in view of ABYSS renders claims 1, 29, and 58 of the '859 patent obvious.

2. The Dependent Claims of the '859 Patent

a. Claims 3, 31, 60 and Claims 5, 33, 62 of the '859 Patent

E152. Claims 3, 31, and 60 depend from claims 1, 29, and 58, respectively, and specify that “said repository comprises means for storing the content.”

E153. Claims 5, 33, and 62 depend from claims 3, 31, and 60, respectively, and specify that the “means for storing” comprises “means for storing content after rendering.”

E154. Rosen shows that the transaction devices each have a storage to store digital content. As explained, during the transaction with the merchant, the customer transaction device will receive and store an encrypted electronic object. Ex. 1020 (Rosen) at 18:60-65, 19: 1-13. See ¶¶ E53-E65, *above*.

E155. It is my opinion that Rosen in view of Hartrick renders claims 3, 31, and 60 and claims 5, 33, and 62 of the '859 patent obvious.

b. Claims 8, 36, 65 and Claims 13, 40, 69 of the '859 Patent

E156. Claims 8, 36, and 65 depend from claims 5, 33, and 62, respectively, and specify that the content to be rendered comprises “video.”

E157. Claims 13, 40, and 69 depend from claims 1, 29, and 58, respectively, and specify that the “rendering device” comprises a “video system.”

E158. Rosen explains that the object content includes video content that the customer can watch. Ex. 1020 (Rosen) at 6:1-2 (“The body is the content which the purchaser can interact with, peruse, or watch.”); *id.* at 4:54-5:3 (discussing multimedia games). Rosen further shows the transactional devices contain specific applications for rendering particular types of content such as movies (*e.g.*, movie, book, multimedia game, etc.). *Id.* at 7:65-8:7, 8:22.

E159. It is my opinion that Rosen in view of Hartrick renders claims 8, 36, and 65 and claims 13, 40, and 69 of the '859 patent obvious.

c. Claims 15, 42, 71 of the '859 Patent

E160. Claims 15, 42, and 71 depend from claims 1, 29, and 58, respectively, and specify that the “rendering device” comprises a “computer system” and the “repository” comprises “software executed on the computer system.”

E161. As I explained above, with respect to repositories, the Rosen transaction devices are computer systems that contain repositories with behavioral integrity, physical integrity, and communications integrity. *See* ¶¶ E136-E145, *above*.. Each of these security requirements are enforced by software running on the trusted agent. *Id.*

E162. It is my opinion that Rosen in view of Hartrick renders claims 15, 42, and 71 of the '859 patent obvious.

d. Claims 16, 43, 72 of the '859 Patent

E163. Claims 16, 43, and 72 depend from claims 1, 29, and 58, respectively, and specify that an “execution engine” is coupled to the repository and the repository will “permit said execution device to execute a computer program only in a manner specified by the usage rights.”

E164. The Rosen electronic objects can include “computer software [and] games,” (Ex. 1020 at 4:55-56), and the CTDs and MTDs contain “processors” used

to “execute an electronic object,” (*id.* at 8:28-31). The usage rights are enforced by the trusted agent on the customer’s device.

E165. It is my opinion that Rosen in view of Hartrick renders claims 16, 43, and 72 of the ’859 patent obvious.

e. **Claims 19, 46, 75 and Claims 20, 47, 76 of the ’859 Patent**

E166. Claims 19, 46, and 75 depend from claims 1, 29, and 58, respectively, and specify that the “manner of use is a manner of displaying.”

E167. Claims 20, 47, and 76 depend from claims 1, 29, and 58, respectively, and specify that the “manner of use is a manner of playing.”

E168. Displaying a work is a manner of “playing” and includes processes such as “playing digital movies, playing digital music, playing a video game, running a computer program, or displaying a document on a display.” Ex. 1001 at 18:49-53.

E169. In the combined system, the MTA would write the Hartrick security and royalty tags in the “Usage” field of a decryption ticket. Ex. 1020 (Rosen) at 6:12-13; Ex. 1021 at 3:53-58, 10:1-14.

E170. As I explained above (¶¶ E85 to E86, E90 to E98), the choice of SGML tags was a simple design choice, and a person of skill could have used various tags, such as a “Display” tag or the “Do Not Copy” tag, to specify a user could only display the electronic object.

E171. That person also would have used, for example, a “Print” tag or the “Do Not Distribute” tag to specify a user could print a paper copy of the electronic object. Displaying a digital work on a monitor or printing are manners of displaying the work and manners of playing the work.

E172. It is my opinion that Rosen in view of Hartrick renders claims 19, 46, and 75 and claims 20, 47, and 76 of the ’859 patent obvious.

f. Claims 21, 48, 77 of the ’859 Patent

E173. Claims 21, 48, and 77 depend from claims 1, 29, and 58, respectively, and specify that “the rendering device and the repository are integrated into a secure system having a secure boundary.”

E174. Each Rosen trusted agent is contained in a physically secure, tamperproof module. *See* Ex. 1020 (Rosen) at 4:4-39, 7:65-8:2. Rosen also explains use of the trusted agents creates a “secure transaction environment for both the buyer and seller of electronic merchandise and services.” *Id.* at 1:6-11, 4:25-32. *See ¶¶ E2-E20, above.*

E175. Each transaction device must contain a trusted agent, which ensures that the transaction device is secure. *Id.* at 7:65-8:7, 4:25-32. Therefore, Rosen shows the “rendering device” and the “repository” are integrated into a secure boundary.

E176. It is my opinion that Rosen in view of Hartrick renders claims 21, 48, and 77 of the '859 patent obvious.

g. Claims 24, 51, 81 of the '859 Patent

E177. Claims 24, 51, and 81 depend from claims 1, 29, and 58, respectively, and specify that the rendering system “comprises means for” or is “configured to” “communicat[ing] with a master repository for obtaining an identification certificate for the repository.”

E178. Each trusted agent must have a valid digital certificate (“*identification certificate*”) issued by the Trusted Agency, which is a certificate authority that issues digital certificate and distributes untrusted lists (“*master repository*”). Ex. 1020 at 11:6-30, 10:45-50. Digital certificates expire, and each trusted agent must periodically connect to the trusted agency to renew its digital certificate. Ex. 1020 at 12:14-33, 15:14-29. *See ¶¶ E28-E37, above.*

E179. Each trusted agent contains a communications interface that allows it to communicate with other trusted agents, including Trusted Agency servers (“*means for . . . communicating with a master repository*”). Ex. 1020 at 9:1-15

E180. Each trusted agent can establish a secure session with a Trusted Agency server and obtain a digital certificate. Ex. 1020 at 12:14-33, 15:15-29.

E181. It is my opinion that Rosen in view of Hartrick renders claims 24, 51, and 81 of the '859 patent obvious.

3. The Claims of the '072 Patent

E182. The independent claims the '072 patent (claims 1, 10, and 18) describe similar processes. Each independent claim specifies a “document platform” that requests and/or receives content and “at least one usage right” from a “document repository.” The document platform stores the content and the usage right in separate files, and it renders the content but only in accordance with the usage right (or rights). Claims 10 and 18 each additionally specify that the document repository authenticates the document platform. Claim 10 additionally specifies that the document repository stores the content and the at least one usage right in separate files.

E183. Rosen discloses the processes described in the '859 independent claims. As explained above, Rosen shows a system for distributing and rendering electronic objects (“digital document”). Ex. 1020 (Rosen) at 1:5-11, 4:55-57. Electronic objects are exchanged using transaction devices (“document repository” and “document platform”), which are computers that contain a trusted agent, a processor, and transaction applications for rendering content. Digital content is stored in electronic objects, and each object is associated with a decryption ticket that contains a usage field for storing usage rights information (“at least one usage right”).

E184. Rosen shows a customer transaction device requesting content (“retrieving . . . a digital document and at least one usage right”) from a merchant transaction device. Rosen shows that the merchant transaction device will receive the request from customer (“receiving a request”), authenticate the customer using a digital certificate, and if the purchase transaction is successful, it will transfer an electronic object and a decryption ticket to the customer (“transferring the digital document”). Rosen shows that the decryption ticket will be stored in the trusted agent, and the encrypted content will be stored in memory (“storing . . . in separate files”). As I explained above (§ E115-E121), it would have been obvious to configure the transaction devices to enforce the usage rights information in the usage field when a user requested to use content.

a. Claim 1 of the '072 Patent

E185. The processes I described above disclose all the limitation of claim 1 of the '072 patent. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'072 Patent: Claim 1</u>	<u>Relevant Concepts and Paragraphs</u>
1. A method for securely rendering digital documents, comprising:	<u>“digital documents”</u> Electronic objects containing softcopy books as described in <u>Hartrick</u> . See §§ E95-E98, E112-E114. <u>“securely rendering digital documents”</u>

	<p>transaction devices have transaction applications for rendering content. ¶¶ E15-E20, E59, E142</p> <p>It would have been obvious to configure the CTA in a customer transaction device to enforce specialized SGML tags stored in the “Usage” field of the decryption ticket. ¶¶ E99-E103, E115-E121</p>
<p>[1.a] retrieving, by a document platform, a digital document and at least one usage right associated with the digital document from a document repository, the at least one usage right specifying a manner of use indicating the manner in which the digital document can be rendered;</p>	<p><u>“retrieving . . . a digital document”</u></p> <p>Customer transaction devices can request content from a merchant. ¶¶ E21-E25, E53-E65, E66-E69, E99-E100</p> <p><u>“document platform”</u></p> <p>The customer transaction devices, each of which contains a trusted agent that is physically secure and that executes secure protocols for acquiring, storing, and rendering content. ¶¶ E2-E10; E24-E37, E136-E145</p> <p><u>“document repository”</u></p> <p>The merchant transaction devices, each of which contains a trusted agent that is physically secure and that executes secure protocols for acquiring, storing, and rendering content. ¶¶ E2-E10; E24-E37, E136-E145</p> <p><u>“at least one usage right”</u></p> <p>It would be obvious to store specialized SGML tags in the “Usage” field of the</p>

	<p>decryption ticket to specify whether content can be displayed, printed, copied, or transferred. ¶¶ E6-E10, E74-E98, E104-E114.</p> <p><u>“a manner of use indicating the manner in which the digital document can be rendered”</u></p> <p>The “Usage” field contains specialized SGML tags, as suggested by <u>Hartrick</u>. ¶¶ E78-E99, E108-E114, E124-E135.</p>
[1.b] storing the digital document and the at least one usage right in separate files in the document platform;	<p>The content is stored in the electronic object, stored on any memory on the transaction device. The usage rights are stored in the “Usage” field of the decryption ticket, stored in the CTA. ¶¶ E7-E10, E58-E59, E95-E96, E112-E113.</p>
[1.c] determining, by the document platform, whether the digital document may be rendered based on the at least one usage right; and	<p>A customer uses a transaction application to send a request to the CTAs on the customer transaction device asking to access the decryption ticket and render content. ¶¶ E15-E20, E59, E99-E103, E115-E121, E142.</p>
[1.d] if the at least one usage right allows the digital document to be rendered on the document platform, rendering the digital document by the document platform.	<p>A customer uses a transaction application to send a request to the CTAs on the customer transaction device asking to access the decryption ticket and render content. ¶¶ E15-E20, E59, E99-E103, E115-E121, E142</p>

b. Claim 10 of the '072 Patent

E186. The processes I described above disclose all the limitation of claim 10 of the '072 patent. I note that Claim 10.b specifies a document repository receives a “request for access” to content and in response it “transfers” content to the document platform. Claims 1 and 18 each specify a “request to transfer.” A person of ordinary skill would understand that a “request for access” to content would be satisfied by a “request to transfer” content. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'072 Patent: Claim 10</u>	<u>Relevant Concepts and Paragraphs</u>
10. A method for securely rendering digital documents, comprising:	<i>See</i> Claim 1.
[10.a] storing a digital document and at least one usage right in separate files in a document repository, wherein the at least one usage right is associated with the digital document;	<p><u>“document repository”</u></p> <p>The merchant transaction devices, each of which contains a trusted agent that is physically secure and that executes secure protocols for acquiring, storing, and rendering content. ¶¶ E2-E10; E24-E37, E136-E145</p> <p><u>“separate files” & “at least one usage right”</u></p> <p>It would have been obvious to configure the MTA to store a decryption ticket with usage information for each copy of content received from an author. ¶¶ E119-E121</p>
[10.b] receiving a request from a document platform for access to the	<u>“request”</u>

<p>digital document;</p>	<p>Customer transaction devices can request content from a merchant. ¶¶ E21-E25, E53-E65, E66-E69, E99-E100, E186.</p> <p><u>“document platform”</u></p> <p>See Claim [1.a].</p>
<p>[10.c] determining, by the document platform, whether the request may be granted based on the at least one usage right, the determining step including authenticating the document platform and determining whether the at least one usage right includes a manner of use that allows transfer of the digital document to the document platform;</p>	<p>See Claim 1.c.</p> <p><u>“authenticating the document platform”</u> merchant transaction devices “authenticate” a customer device. ¶¶ E4, E28-E37, E39-E44, E123</p> <p><u>“a manner of use that allows transfer”</u></p> <p>When a CTA receives an electronic object and a decryption ticket, the CTA will validate the object’s integrity and ensure it is the requested object, and it would have been obvious to also validate the “Usage” field during this process. ¶ E58</p> <p>It would have been obvious to have authors specify usage information when depositing content with merchants, and to configure the MTA in a merchant transaction device to enforce specialized SGML tags stored in the “Usage” field of the decryption ticket. ¶¶ E99-E103, E115-E121</p>
<p>[10.d] if the at least one usage right allows the transfer of the digital document to the document platform,</p>	<p>It would have been obvious to have authors specify usage information when depositing content with merchants, and</p>

transferring the digital document and the at least one usage right associated with the digital document to the document platform;	to configure the MTA in a merchant transaction device to enforce specialized SGML tags stored in the “Usage” field of the decryption ticket. ¶¶ E99-E103, E115-E121. In response to the purchase request, the MTA will generate a new electronic object and new decryption ticket and transmits those to the CTA. ¶¶ E57, E119-E121
[10.e] storing the digital document and the at least one usage right in the document platform, wherein the at least one usage right is stored in a separate file from the digital document; and	See Claim 1.b.
[10.f] rendering the digital document by the document platform.	See Claim 1.d.

c. Claim 18 of the '072 Patent

E187. The processes I described above disclose all the limitation of claim 18 of the '072 patent. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'072 Patent: Claim 18</u>	<u>Relevant Concepts and Paragraphs</u>
18. A method for securely rendering digital documents, comprising:	See Claim 1.
[18.a.] receiving a request from a document platform to transfer a digital document from a document repository to the document platform, the digital document having at least one usage right associated therewith;	See Claim 10.b. <u>“document platform”</u> The customer transaction devices, each of which contains a trusted agent that is physically secure and that executes

	<p>secure protocols for acquiring, storing, and rendering content. ¶¶ E2-E10; E24-E37, E136-E145</p> <p><u>“receiving a request . . . to transfer a digital document”</u></p> <p>Customer transaction devices can request content from a merchant. ¶¶ E21-E25, E53-E65, E66-E69, E99-E100</p> <p><u>“document repository”</u></p> <p>The merchant transaction devices, each of which contains a trusted agent that is physically secure and that executes secure protocols for acquiring, storing, and rendering content. ¶¶ E2-E10; E24-E37, E136-E145</p> <p><u>“at least one usage right”</u></p> <p>See Claim [10.a].</p>
<p>[18.b.] authenticating the document platform by accessing at least one identifier associated with the document platform or with a user of the document platform and determining whether the identifier is associated with at least one of the document platform and a user authorized to use the digital document;</p>	<p>See Claim 1.c & 10.c.</p> <p><u>“identifier associated with the document platform or with a user”</u></p> <p>Each trusted agent has a digital certificate issued by the “Trusted Agency,” and the certificate contains the trusted agent’s id number. ¶¶ E29-E33, E43; <i>see generally</i> E24-E37</p> <p><u>“accessing”</u></p> <p>The MTA validates the digital</p>

	<p>certificate. ¶¶ E29-E33, E43.</p> <p><u>“determining whether the identifier is associated with at least one [entity] authorized to use the digital document”</u></p> <p>The MTA validates (“accessing”) the digital certificate and verifies that the CTA’s id is not on the untrusted list ¶¶ E29-E33, E43; <i>see generally</i> E24-E37</p>
<p>[18.c.] if the authenticating step is successful, determining whether the digital document may be transferred and stored on the document platform based on a manner of use included in the at least one usage right;</p>	<p><u>“the authenticating step [was] successful”</u></p> <p>Establishing the session between the CTA and MTA ¶¶ E38-E52 (transfers)</p> <p><u>“determining whether the digital document may be transferred and stored on the document platform based on a manner of use included in the at least one usage right”</u></p> <p>It would have been obvious to have authors specify usage information when depositing content with merchants, and to configure the MTA in a merchant transaction device to enforce specialized SGML tags stored in the “Usage” field of the decryption ticket. ¶¶ E99-E103, E115-E121.</p>
<p>[18.d.] if the at least one usage right allows the digital document to be transferred to the document platform, transferring the digital document and the</p>	<p><i>See</i> Claim 10.d.</p>

at least one usage right associated with the digital document to the document platform;	
[18.e.] storing the digital document and the at least one usage right in the document platform, wherein the at least one usage right is stored in a separate file from the digital document;	See Claim 10.e.
[18.f.] determining, by the document platform, whether the digital document may be rendered based on the at least one usage right; and	See Claim 1.c.
[18.g.] if the at least one usage right allows the digital document to be rendered on the document platform, rendering the digital document by the document platform	See Claim 1.d.

E188. It is my opinion that Rosen in view of ABYSS renders claims 1, 10, and 18 of the '072 patent obvious.

d. Claims 8, 16, and 24 of the '072 Patent

E189. Claims 8, 16, and 24 depend from claims 1, 10, and 18, respectively, and specify that “at least one part of the digital document and the at least one usage right are stored on a same device.”

E190. Both the digital document and the usage rights can be stored within the trusted agent of Rosen. Rosen explains that the purchased “merchandise” can be stored “within [the] trusted agent. See Ex. 1020 (Rosen) at 19:40-43; 23:17-21 (explaining that the digital document is provisionally retained within the trusted agent). This merchandise includes both the digital document (electronic object) as

well as the usage rights (decryption tickets). *Id.* at 17:43-48. *See* ¶¶ E53-E65, *above*.

E191. It is my opinion that Rosen in view of Hartrick renders claims 8, 16, and 24 of the '072 patent obvious.

4. The Claims of the '956 and '007 Patents

E192. I am addressing the claims of the '956 and '007 patents together. Both the '956 and '007 claims are directed to the same basic processes for transferring content between a sending device and recipient device. The '956 claims are directed toward client or recipient device, while the '007 claims are directed toward the server or sending device.

a. Claims 1, 7, and 13 ('956) and 1, 6, and 11 ('007)

E193. Rosen discloses the processes described in the '956 and '007 independent claims. Because the independent claims are highly similar, I have addressed them together below. In the tables below, I have mapped each claim element to the relevant paragraphs of my analysis.

E194. Rosen explains that a customer has a transaction device containing a trusted agent (“recipient computing device” or “apparatus”) and each merchant has a transaction device containing a trusted agent (“sending computing device”). A customer can purchase and retrieve an electronic object (“content”) and a decryption ticket containing a usage field (“usage rights information”) from the

merchant. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
<p>[Claim 1. preamble] A computer-implemented method of rendering digital content by at least one recipient computing device in accordance with usage rights information, the method comprising:</p>	<p>[Claim 1. preamble] A computer-implemented method of distributing digital content to at least one recipient computing device to be rendered by the at least one recipient computing device in accordance with usage rights information, the method comprising:</p>	<p><u>“recipient computing device”</u></p> <p>The customer transaction devices, each of which contains a trusted agent that is physically secure and that executes secure protocols for acquiring, storing, and rendering content. ¶¶ E2-E10; E24-E37, E136-E145</p> <p><u>“sending computing device”</u></p> <p>The merchant transaction devices, each of which contains a trusted agent that is physically secure and that executes secure protocols for acquiring, storing, and rendering content. ¶¶ E2-E10; E24-E37, E136-E145</p>

		<p><u>“digital content”</u></p> <p>Electronic object. ¶¶ E21-E27;</p> <p><u>“usage rights information”</u></p> <p>It would be obvious to store specialized SGML tags in the “Usage” field of the decryption ticket to specify whether content can be displayed, printed, copied, or transferred. ¶¶ E6-E10, E74-E98, E104-E114.</p>
<p>[Claim 7. preamble] A recipient apparatus for rendering digital content in accordance with usage rights information, the recipient apparatus comprising:</p>	<p>[Claim 6. preamble] A sending apparatus for distributing digital content to at least one recipient computing device to be rendered by the at least one recipient computing device in accordance with usage rights information, the sending apparatus comprising:</p>	<p>See claim 1, above</p>
<p>[Claim 13. preamble] At least one non-transitory computer-readable medium storing computer-readable instructions that, when</p>	<p>[Claim 11. preamble] At least one non-transitory computer-readable medium storing computer-readable instructions that, when</p>	<p>See claim 1, above</p>

executed by at least one recipient computing device, cause the at least one recipient computing device to:	executed by at least one sending computing device, cause the at least one sending computing device to:	
--	--	--

E195. Rosen discloses systems, methods, and software for implementing the techniques described in the patent. As I explained above, each transaction device contains a processor, memory, and software for enabling its functionality. *See* ¶¶ E11-E20, *above*. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
<p>[7.a.] one or more processors; and</p> <p>[7.b.] one or more memories operatively coupled to at least one of the one or more processors and having instructions stored thereon that, when executed by at least one of the one or more processors, cause at least one of the one or more processors to:</p>	<p>[6.a.] one or more processors; and</p> <p>[6.b.] one or more memories operatively coupled to at least one of the one or more processors and having instructions stored thereon that, when executed by at least one of the one or more processors, cause at least one of the one or more processors to:</p>	<p>Each Rosen transaction device has a processor and memory. ¶¶ E11-E20</p> <p>Each Rosen trusted agent has a processor and memory. ¶¶ E2-E10</p>

E196. Rosen explains that, when a client transaction device requests to purchase content from a merchant transaction device, the merchant transaction

device will authenticate the customer by validating the customer’s digital certificate (“determining if the . . . recipient computing device is trusted to receive the digital content”). If the client has a valid and unrevoked certificate, the merchant will establish a session with the client to allow the client to purchase the content. If the purchase transaction is successful, the merchant will transfer an electronic object containing the content and a decryption ticket to the customer (“receiving the digital content”). In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[1.a.] receiving the digital content by the at least one recipient computing device from at least one sending computing device only if the at least one recipient computing device has been determined to be trusted to receive the digital content from the at least one sending computing device;	[1.a.] determining, by at least one sending computing device, if the at least one recipient computing device is trusted to receive the digital content from the at least one sending computing device;	<p><u>“determining if the . . . recipient computing device is trusted to receive the digital content”</u></p> <p>The MTA authenticates the CTA by validating its digital certificate. ¶¶ E4, E24-E37, E39-E45</p> <p><u>“receiving digital content”</u></p> <p>Customer transaction devices can request and receive content from a merchant. ¶¶ E21-E25, E53-E65, E66-E69, E99-E100, E186.</p>

[7.c.] enable the receipt of the digital content by the recipient apparatus from at least one sending computing device only if the recipient apparatus has been determined to be trusted to receive the digital content from the at least one sending computing device;	[6.c.] determine if the at least one recipient computing device is trusted to receive the digital content from the sending apparatus;	<i>See claim 1, above.</i>
[13.a.] receive the digital content from at least one sending computing device only if the at least one recipient computing device has been determined to be trusted to receive the digital content from the at least one sending computing device;	[11.a.] determine if the at least one recipient computing device is trusted to receive the digital content from the at least one sending computing device;	<i>See claim 1, above.</i>

E197. As I just explained, if merchant determines the customer has a valid certificate and the purchase transaction is successful, the merchant will transfer an electronic object containing the content (“sending the digital content”) and a decryption ticket to the customer (“sending the usage information”). In the table

below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
N/A	<p>[1.b.] [11.b.] send[ing] the digital content, by the at least one sending computing device, to the at least one recipient computing device only if the at least one recipient computing device has been determined to be trusted to receive the digital content from the at least one sending computing device; and</p> <p>[6.d.] send the digital content, by the sending apparatus, to the at least one recipient computing device only if the at least one recipient computing device has been determined to be trusted to receive the digital content from the sending apparatus; and</p>	<p><i>See ¶¶ 1.a, above.</i></p> <p>In response to the purchase request, the MTA will generate a new electronic object and new decryption ticket and transmits those to the CTA. ¶¶ E57, E119-E121</p>
	<p>[1.c.] [11.c.] send[ing] usage rights information indicating how the digital content may be rendered by the at least one recipient computing device, the usage rights information being enforceable by the at least on recipient computing device.</p> <p>[6.e.] send usage rights information indicating how the digital content may be rendered by the at least one recipient computing device, the usage rights information being enforceable by the at</p>	<p>In response to the purchase request, the MTA will generate a new electronic object and new decryption ticket and transmits those to the CTA. ¶¶ E57, E119-E121</p> <p>It would be obvious to store specialized SGML tags in the “Usage” field of the</p>

	least on recipient computing device.	decryption ticket to specify whether content can be displayed, printed, copied, or transferred. ¶¶ E6-E10, E74-E98, E104-E114.
--	--------------------------------------	--

E198. Rosen explains that transaction devices have applications for rendering content (“rendering the digital content”). For a user to render content, it necessarily must request a transaction device to render the content (“receiving a request . . . to render the digital content”).

E199. As I explained above, it also would have been obvious based on Hartrick to configure the transaction applications to enforce the usage field of a decryption ticket before permitting allowing a user to render content (“determining . . . whether the digital content may be rendered”).

E200. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[1.b.], [7.d], [13.b] receiv[ing], [by the at least one recipient computing device,] a request to render the digital content;	N/A	<p><u>“rendering the digital content”</u></p> <p>Rosen transaction devices have applications for rendering content. ¶¶ E20; E71-E103</p> <p><u>“receiving a request .</u></p>

		<u>.. to render the digital content.”</u> A user requests the transaction device to render the content. ¶¶ E75; E82-E103
[1.c.], [7.e], [13.c] determin[ing], based on the usage rights information, whether the digital content may be rendered by the at least one recipient computing device [apparatus]; and		See claim 1, above
[1.d.], [7.f], [13.d] render[ing] the digital content, [by the at least one recipient computing device,] only if it is determined that the content may be rendered by the at least one recipient computing device [apparatus].		See claim 1, above

E201. Therefore, it is my opinion that Rosen in view of Hartrick renders claims 1, 7, and 13 of the '956 patent obvious, as well as claims 1, 6, and 11 of the '007 patent.

b. Claims 2, 8, 14 ('956)

E202. Claims 2, 8, and 14 of the '956 depend from claims 1, 7, and 13, respectively, and each specifies that the recipient device will “*deny[] the request and prevent[] rendering of the digital content . . . if it is determined that the digital content may not be rendered . . .*”

E203. As I explained above, it also would have been obvious based on Hartrick to configure the transaction applications to enforce the usage field of a

decryption ticket before permitting allowing a user to render content (“determining . . . whether the digital content may be rendered”). Therefore, it is my opinion that Rosen in view of Linn renders claims 2, 8, and 14 of the ’956 patent obvious.

c. Claims 3, 9, 15 (’956) & 2, 7, 14 (007)

E204. Claims 3, 9, and 15 of the ’956 depend from claims 1, 7, and 13, respectively, and each specifies that the recipient device will “*deny[] the request and prevent[] rendering of the digital content . . . if it is determined that the digital content may not be rendered . . .*”

E205. Claims 2, 7, and 14 of the ’007 depend from claims 1, 6, and 11, respectively, and each specifies the usage rights information includes a “*condition under which the content can be rendered.*”

E206. As I explained above, it also would have been obvious based on Hartrick to configure the trusted agents to enforce the usage field of a decryption ticket before permitting a user to access a decryption ticket and render content (“determining . . . whether the digital content may be rendered”). Ex. 1021 at 6:9-22, Fig. 1. Thus, if the restrictions in the “Usage” field did not specify the requested use was permitted, the CTA would deny the request and prevent the application from rendering the content. Ex. 1021 at 15:23-40, 16:10-22. In a combined Rosen/Hartrick system, the “Usage” field could contain royalty tags,

such as a “print-royalty” tag that specified an additional payment must be made to exercise the print right. *See* ¶¶ E87-E89, E90-E98, *above*.

E207. Therefore, it is my opinion that Rosen in view of Hartrick renders claims 3, 9, and 15 of the ’956 patent obvious, as well as claims 2, 7, and 14 of the ’007 patent..

d. Claims 4, 10, 16 (’956) and 3, 8, 16 (’007)

E208. Claims 4, 10, and 16 of the ’956 depend from claims 1, 7, and 13, respectively, and each specifies that the step of “*receiving the digital content comprises*” two additional steps by the recipient device: requesting an “authorization object” and “receiving” it if authorized.

E209. Claims 3, 8, and 16 of the ’007 depend from claims 1, 6, and 11, respectively, each and specifies that the step of making “*the determination of trust*” comprises two additional steps by the sending device: receiving a “request . . . for an authorization object” and transmitting the object “when it is determined that the request should be granted.”

E210. As I explained with respect to the purchase transaction above (¶¶ E56, E62), the Rosen MTA can be configured to require a customer to present a payment credential (*e.g.*, a credit card credential) before it will transmit an electronic object and decryption ticket to the customer. Ex. 1020 at 5:4-7, 24:21-

49. Without a valid payment credential, a customer cannot receive, and therefore cannot use, an electronic object. Ex. 1020 at 24:26-29; *see* ¶ E60, *above*.

E211. Each credential ticket is a file of standard format; Rosen explains each “credential ticket” contains fields such as the owner’s name, signature, and an identification number. Ex. 1020 at 6:14-25, Fig. 2, 5:38-58; *see* ¶ E27, *above*.

E212. Each credential ticket also contains “*usage rights*”: each has a “*manner of use*,” such as a flag indicating whether it can be used (“In Use field 60” in Rosen)), and each has a “condition,” such as an expiration date (Ex. 1020 at 6:17-18). *Id.* at 6:19-22 (“an In Use field 60 indicating when a copy of the ticket has been presented to a MTA 4 for use, so that [it] cannot be reused during [] presentation”); *see id.* at Fig. 2, 6:14-25.

E213. Therefore, the Rosen credential ticket is an “*authorization object*”, *i.e.*, “a digital work in a file of a standard format” that is used “to receive the digital content and to use the digital content.”

E214. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[4.a.], [10.a.], [16.a.] requesting an authorization object for the at least one recipient computing device	[3.a.], [8.a.], [16.a.] receiving a request from at least one recipient computing device for an	<u>“requesting an authorization object”/</u> <u>“receiving a request [for an authorization object”</u>

<p>[receiving apparatus] to make the digital content available for use, the authorization object being required to receive the digital content and to use the digital content; and</p>	<p>authorization object required to render the digital content; and</p>	<p>CTA must request and receive a payment credential ticket from an ATA through a ticket purchase transaction. ¶¶ E60, E66-E69</p> <p>It would have been obvious to incorporate the <u>Hartrick</u> authorization message into the <u>Rosen</u> system. To exercise rights for a royalty fee, the CTA must request an authorization message. ¶¶ E99-E103, E112-E114</p>
<p>[4.b.], [10.b.], [16.b.] receiving the authorization object if it is determined that the request for the authorization object should be granted.</p>	<p>[3.b.], [8.b.], [16.b.] transmitting the authorization object to the at least one recipient computing device when it is determined that the request should be granted.</p>	<p><u>“determin[ing] that a request . . . should be granted”</u></p> <p>The ATA will evaluate whether a request should be granted through a ticket purchase transaction, and if so, it will provide a credential ticket. ¶¶ E66-E69.</p> <p>It would have been obvious to incorporate the <u>Hartrick</u> authorization message into the <u>Rosen</u> system. The MTA will determine whether the request should be granted (<i>e.g.</i>, can the user pay), and if so it will return an authorization message. ¶¶ E99-E103,</p>

		E112-E114
--	--	-----------

E215. Therefore, it is my opinion that Rosen in view of Hartrick renders claims 4, 10, and 16 of the '956 patent obvious. It is my opinion that Rosen in view of Hartrick renders claims 3, 8, and 16 of the '007 patent obvious.

e. Claims 5, 11, 17 ('956) and Claims 4, 9, 14 ('007)

E216. Claims 5, 11, and 17 of the '956 patent depend from claims 1, 7, and 13, respectively, each specifies a method, apparatus, or software where the step of “receiving the digital content” or “enabling the receipt of the digital content” comprises three additional steps by the recipient computing device: generating a “registration message,” exchanging a session key with the sending device, and conducting a session where it receives content from the sending device.

E217. Claims 4, 9, and 14 of the '007 patent depend from claims 1, 6, and 13, respectively, each specifies a method, apparatus, or software where the step of making “the determination of trust” comprises four additional steps by the sending computing device: receiving a “registration message” from a recipient device, validating the authenticity of the recipient device, exchanging a session key with the recipient computing device, and conducting a session where it sends content to the recipient device.

E218. As with the independent claims, these '956 and '007 claims are directed to the same process, but the '956 claims cover the client device and the '007 claims cover the server device. These additional steps simply add standard cryptographic techniques that were well-known in the art.

E219. Rosen shows that when a customer finds content to purchase, the customer sends a request to the merchant. Each transaction device passes the process to its trusted agent. Rosen indicates that next the customer trusted agent will initiate the Establish Session protocol by sending a request to the merchant trusted agent. As I explained above, it would have been obvious to configure this process so that the merchant transaction device initiated the Establish Session protocol. *See* ¶¶ E48-E51, A148-A153, *above*. In that configuration, the MTA would send the CTA a message containing the MTA's digital certificate. The CTA validates the certificate, and if valid, the CTA responds by sending the MTA a message containing the CTA's certificate, a first random number ("exchanging messages including . . . [a] session key"), a first "verification message" (a "random registration identifier"), and the date and time. The MTA validates the certificate ("validating the [device's] authenticity"), and if valid, the MTA generates a second random number ("exchanging messages including . . . [a] session key") and second verification message (a "nonce") and sends them, the first verification number, and the date and time to the CTA. The CTA validates the message, sends an

acknowledgement containing the second verification message. The MTA validates the verification message is correct and then the secure channel is established.

E220. After the session is established, the merchant trusted agent will validate the customer's payment information ("conducting a secure transaction using the session key"). If valid, the merchant trusted agent will transfer an electronic object and a decryption ticket to the customer trusted agent using the encrypted connection ("sending [receiving] the digital content").

E221. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
[5.a.], [11.a.], [17.a.] generating a registration message, the registration message including an identification certificate of the recipient computing device [apparatus] and a random registration identifier, the identification certificate being certified by a master device;	[4.a.] [9.a.] [14.a.] receiving a registration message from the at least one recipient device, the registration message including an identification certificate of the recipient computing device and a random registration identifier, the identification certificate being certified by a master device;	<p><u>"identification certificate"</u></p> <p>Digital certificate. Ex. ¶¶ E24-E37</p> <p><u>"registration message"</u></p> <p>CTA sends the MTA a message. ¶¶ E219, E42-E47</p> <p><u>"registration identifier"</u></p> <p>First verification message. ¶¶ E219, E43, E47</p>

		<p><u>“nonce”</u></p> <p>Second verification message. ¶¶ E219, E44-E47.</p> <p><u>“master device”</u></p> <p>Trusted Agency. ¶¶ E5, E12, E27-E37</p>
N/A	[4.b.] [9.b.] [14.b.] validating the authenticity of the at least one recipient device;	The MTA validates the CTA’s digital certificate. ¶¶ E219, E38-E51
[5.b.], [11.b.], [17.b.] exchanging messages including at least one session key with at least one provider computing device, the session key to be used in communications during a session; and	[4.c.] [9.c.] [14.c.] exchanging messages including at least one session key with the at least one recipient device, the session key to be used in communications; and	The exchanged messages will include random numbers that are used to generate the session key. ¶¶ E38-E52
[5.c.], [11.c.], [17.c.] conducting a secure transaction using the session key, wherein the secure transaction includes receiving the digital content.	[4.d.] [9.d.] [14.d.] conducting a secure transaction using the session key, wherein the secure transaction includes sending the digital content to the at least one recipient device.	After the session is established, the MTA will transmit the digital content and decryption ticket to the CTA through the encrypted channel. ¶¶ E53-E65

E222. It is my opinion that Rosen in view of Hartrick renders claims 5, 11, and 17 of the '956 patent obvious. It is my opinion that Rosen in view of Hartrick renders claims 4, 9, and 14 of the '007 patent obvious.

f. Claims 6, 12, 18 ('956) and Claims 5, 10, 15 ('007)

E223. Claims 6, 12, and 18 of the '956 patent depend from claims 5, 11, and 17, respectively, each specifies the method, apparatus, or software comprises two additional steps. The two additional steps in each of claims 6, 12, and 18 are identical. These steps simply add standard cryptographic techniques that were well-known in the art.

E224. Claims 5, 10, and 15 of the '007 patent depend from claims 1, 9, and 14, respectively, each specifies a method, apparatus, or software where the step of making “validating” comprises four additional steps. The four additional steps in each of claims 5, 10, and 15 are identical.

E225. As with the independent claims, these '956 and '007 claims are directed to the same process, but the '956 claims cover the client device and the '007 claims cover the server device. These additional steps simply add standard cryptographic techniques that were well-known in the art.

E226. As I explained in ¶ 219, it would be obvious to configure Rosen so the MTA sent the first message of the Establish Session protocol. In that configuration, the MTA would send the CTA its digital certificate, and the CTA

would respond by sending the MTA a message containing the CTA's certificate, a first random number, a first "verification message", and the date and time. The MTA validates the certificate ("verifying the identification certificate"), and if valid, the MTA generates ("generating a message...") a second random number and second verification message (a "nonce") and sends them, the first verification number, and the date and time to the CTA ("receiving a message to test the authenticity").

E227. In the next step, the CTA validates the message from the MTA, sends an acknowledgement containing the second verification message ("processing to indicate authenticity"). The MTA validates the verification message is correct ("verifying the [device's] authenticity") and then the secure channel is established.

E228. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'956 Patent</u>	<u>'007 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
N/A	[5.a.] [10.a.] [15.a.] verifying the identification certificate of the at least one recipient device	The MTA validates the CTA's digital certificate.
[6.a.], [12.a.], [18.a.] receiving a message to test the authenticity of the at least one recipient computing device [apparatus], the generated	[5.b.] [10.b.] [15.b.] generating a message to test the authenticity of the at least one recipient device, the generated message including a nonce;	The verification message contains a nonce. ¶¶ E45-E52

message including a nonce; and	[5.c.] [10.c.] [15.c.] sending the generated message to the at least one recipient device; and	
[6.b.], [12.b.], [18.b.] processing the generated message to indicate authenticity.	[5.d.] [10.d.] [15.d.] verifying if the at least one recipient device correctly processed the generated message.	The CTA will return the verification message to the MTA along with other data and the MTA will check the verification message to make sure it is correct. ¶¶ E47; <i>see generally</i> E38-E52

E229. It is my opinion that Rosen in view of Hartrick renders claims 6, 12, and 18 of the '956 patent obvious. It is my opinion that Rosen in view of Hartrick renders claims 5, 10, and 15 of the '007 patent obvious.

5. Analysis of the Claims of the '576 Patent

a. Claim 1 of the '576 Patent

E230. Claim 1 of the '576 patent contains several means-plus-function elements. I set forth the constructions of these terms that I am applying in ¶¶ B170-B178, above.

E231. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'576 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
--------------------	---

<p>1. An apparatus for rendering digital content in accordance with rights that are enforced by the apparatus, said apparatus comprising:</p>	<p><u>Rosen</u> explains a customer has a CTD (“<i>apparatus</i>”) containing a CTA that stores purchased electronic objects (“<i>digital content</i>”) and corresponding decryption tickets. <i>See</i> Ex. 1020 at 7:65-8:53 (transaction device); 19:15-39 (describing storage of transferred electronic object and decryption ticket), 24:52-55 (rendering), Fig. 2.</p> <p>The <u>Rosen</u> CTD has transaction applications for rendering content, and, as I explained above (¶¶ E98-E103, E104-E122) a person of ordinary skill would have configured the CTA to enforce the “Usage” field of the corresponding decryption ticket when rendering content (“<i>rendering digital content in accordance with rights that are enforced by the apparatus</i>”).</p>
<p>[1.a.] a rendering engine configured to render digital content;</p>	<p><u>Rosen</u> shows the CTD contains “processors” (“<i>rendering engine</i>”) used to “execute an electronic object,” (Ex. 1020 at 8:28-31).</p> <p>These processors are suitable to the “type of electronic object (e.g., movie, book, multimedia game, etc.),” (<i>id.</i> at 4:55-56, 7:65-8:7, 8:22-34).</p>
<p>[1.b.] a storage for storing the digital content;</p>	<p><u>Rosen</u> shows the electronic object can be stored in the CTA or elsewhere in the memory (“<i>a storage</i>”) in the CTD because it is encrypted, and the content cannot be accessed unless the trusted agent allows access to the decryption ticket. <i>Id.</i> at 19:43, 23:16-21, 4:65-5:3.</p>
<p>[1.c.] means for requesting use of the digital content stored in the storage; and</p>	<p><u>Rosen</u> explains teach transaction device has a “Human/Machine Interface</p>

	<p>function” that “provides the look and feel of the transaction device” and can “include a keyboard, mouse, pen, voice, touch screen, icons, menus, etc.” Ex. 1020 at 8:35-38; <i>see</i> ¶¶ E16-E20, <i>above</i>.</p> <p>Requests placed through the Human/Machine Interface are relayed by a Message Manager, that can, <i>inter alia</i>, route the message requesting execution of stored electronic content (“<i>requesting use of the digital content</i>”) to relevant the transaction application. <i>See id.</i> at 8:26-34, 8:38-43, 8:49-52.</p>
[1.d.] a repository coupled to the rendering engine,	<p><u>Repository</u></p> <p>Transaction devices contain trusted agents. ¶¶ E2-E10; E24-E37, E136-E145</p>
[1.e.] wherein the repository includes:	See [1.d]
<p>[1.f.] means for processing a request from the means for requesting,</p> <p><u>This corresponds to the following algorithm:</u></p> <p>Processing begins when a requester repository has sent a message to initiate a request. (Ex. [1009], 36:35-40, 37:9-13.) [1], the server repository checks the compatibility of the requester repository and the validity of the requester’s identification. ([<i>Id.</i>], 36:41-44, 37:14-17.) [2], the requester and server repositories perform the common opening transaction steps of determining</p>	<p>In the combined <u>Rosen/Hartrick</u> system, when a customer purchases content from a merchant, the CTD sends a request to the MTD (Ex. 1020 at 8:22-34, 17:49-60), and the CTA initiates the establish session protocol with the MTA (step 1).</p> <p>After the session is established, the MTA will check the “Usage” field of the decryption ticket to ensure the sale is consistent with author’s predetermined restrictions and any required fees are paid (step 2).</p> <p>The MTA will send the content to the</p>

<p>whether authorization is needed, and whether the requested transaction is permitted given the usage rights. ([<i>Id.</i>], 36:45-46, 37:18-19, 30:59-31:49.) [3], [they] perform a transmission protocol to read and write blocks of data, and then the requester repository renders the digital work. ([<i>Id.</i>], 36:47- 50, 37:20-22.) [4], the contents are removed from the rendering device and the requester repository. ([<i>Id.</i>], 36:51-52, 37:23-24.) [5], [they] perform the common closing transaction steps of updating the usage rights and billing information. ([<i>Id.</i>], 36:53-54, 37:25-26, 32:20-31.)</p>	<p>CTA using <u>Rosen</u>'s two-phase commit transfer process. <i>See id.</i> at 13:37-15:11.</p> <p><u>Rosen</u> teaches the two-phase commit process ensures an electronic object is successfully received by one device and removed from the other device (step 3). <i>Id.</i> at 23:9-51, 13:41-46, 14:12-37 (describing the "Abort and Commit" protocols).</p> <p>If the electronic object is accompanied by a single performance right (<i>e.g.</i>, a movie, <i>id.</i> at 4:56, 5:63-64) or a single print right, a person of ordinary skill would have understood that a logical modification to the combined system would allow the electronic object and corresponding ticket to be deleted once the user exhausted the right (step 4).</p> <p>Finally, <u>Rosen</u>'s trusted agents also record the transactions in the transaction log so that transactions can be properly tracked and billed (step 5). <i>Id.</i> at 9:22-33, 23:46-51.</p>
<p>[1.g.] means for checking whether the request is for a permitted rendering of the digital content in accordance with rights specified in the apparatus,</p> <p><u>This corresponds to the following algorithm:</u></p> <p>[1], the requester repository determines whether an authorization certificate or a digital ticket is needed. (Ex. [1009], 31:3-9.) [2], the server repository</p>	<p>In the combined <u>Rosen/Hartrick</u> system, when the royalty checking protocol on the CTA intercepts a request for use, the CTA will determine if additional fees need to be paid (step 1). Ex. 1020 at 24:52-55; Ex. 1021 at 3:31-40.</p> <p>If the user has chosen an action that requires payment of an additional royalty (<i>e.g.</i>, the user chose to print, and there is a "print-royalty" tag specifying a fee for that action), the CTA will send</p>

<p>generates a transaction identifier. ([<i>Id.</i>], 31:10-13.) [3] the server repository determines whether the right is granted and whether time, security, and access based conditions are satisfied. ([<i>Id.</i>], 31:13-33) [4], the server repository determines whether there are sufficient copies of the work to distribute. ([<i>Id.</i>], 31:34-49.)</p>	<p>a request for authorization to the MTA. Ex. 1021 at 15:34-40, 15:53-16:14, 16:22-25; 21:67-22:3.</p> <p>The MTA receives the request records it for billing purposes (step 2) and determine if the user can pay the required fee for the request and it is otherwise permitted (e.g., copies are available) (steps 3 & 4). <i>See</i> Ex. 1021 at 15:34-40, 15:53-16:14, 16:22-25; 21:67-22:3.</p>
<p>[1.h.] means for processing the request to make the digital content available to the rendering engine for rendering when the request is for a permitted rendering of the digital; and</p> <p><u>This corresponds to the following algorithm:</u></p> <p>[1] transaction information is provided to the server repository by the requester repository. (Ex. [1009], 33:3-8.) [2], the server repository transmits a block of data to the requester repository and waits for an acknowledgement provided by the requester when the block of data has been received. ([<i>id.</i>], 33:9-15.) Unless a communications failure terminates the transaction, that process repeats until there are no more blocks to transmit. ([<i>id.</i>], 33:16-38, 33:46-49.) [3], the requester repository commits the transaction and sends an acknowledgement to the server repository. ([<i>id.</i>], 33:39-45.)</p>	<p>During the <u>Rosen</u> purchase transaction, the CTA provides validation and payment information (step 1), after which the MTA sends content to the CTA (step 2). Ex. 1020 at 19:13-30; <i>see</i> Ex. 1021 at 16:22-37.</p> <p><u>Hartrick</u> teaches an acknowledgement signal periodically sent to the publisher when the user is rendering content to ensure the user is only charged for accessed content. Ex. 1021 at 16:37-52.</p> <p>A person of ordinary skill would seek to incorporate this element into transfer of content on the combined system as a fail-safe to ensure that the user receives all the content paid for (both <u>Hartrick</u> and <u>Rosen</u> disclose means of recording aborted transactions) (step 2). Ex. 1020 at 14:52-15:11 (describing aborting a transaction); Ex. 1021 at 16:34-17:5 (describing periodic acknowledgement signals and recording partial transfers).</p> <p>If the transaction is successful, the CTA</p>

	<p>and the MTA would commit and send each other acknowledgements, just as with other <u>Rosen</u> transactions (step 3). <i>See</i> Ex. 1020 at 14:47-51, 16:52-58, 23:41-51.</p>
<p>[1.i.] means for authorizing the repository for making the digital content available for rendering, wherein the digital content can be made available for rendering only by an authorized repository, the repository comprising:</p> <p><u>This corresponds to the following algorithm:</u></p> <p>[1], a communication channel is set up between the server and the remote repository. (Ex. [1009], 41:52-54.) [2], the server repository performs a registration process with the remote repository. ([<i>id.</i>], 41:55-57.) [3], the server repository invokes a “play” transaction to acquire the authorization object. ([<i>id.</i>], 41:58-64.) [4], the server repository performs tests on the authorization object or executes a script before signaling that authorization has been granted for rendering content. ([<i>id.</i>], 41:65-42:16.)</p>	<p><u>First</u>, when the user requests to render content, the rights checking protocol intercepts the request and identifies a fee that needs to be paid prior to permitting rendering. Ex. 1021 at 6:9-22.</p> <p>If “the user selects to pay the royalty, a communication session is established between” the CTA and MTA (step 1). <i>See id.</i>; Ex. 1021 at 6:34-38.</p> <p>The CTA sends an authorization request containing, <i>inter alia</i>, payment and personal information (step 2). Ex. 1021 at 6:38-49.</p> <p>The MTA processes the request (step 3) and will return an authorization message containing, <i>inter alia</i>, authentication data that the CTA can use to authenticate the authorization message prior to allowing rendering (step 4). <i>See id.</i> at 6:49-7:10</p> <p><u>Second</u>, a user may need a credential ticket (containing personal identification information) to make purchases. To obtain the credential or renew it, a CTA will send a request or revalidation message, respectively, and establish a secure session with an Identification Authority trusted agent (steps 1 & 2). Ex. 1020 at 26:65-67; Fig. 28.</p>

	<p>The Identification Authority trusted agent will check if the CTA’s credential can be validated (step 3), and if so will send the credential to the CTA, which will return an acknowledgement (step 4). <i>Id.</i> at 27:62-28:5, Fig. 28.</p> <p><u>Third</u>, to renew its digital certificate, <u>Rosen</u> explains that the trusted agent will establish “a cryptographically secure communication channel between [it] and the trusted server,” (step 1) (Ex. 1020 at 15:21-23) and send a request to a trusted server to obtain a new digital certificate (step 2) (<i>id.</i> at 12:14-33, 15:15-29).</p> <p>This request begins a message exchange in which the trusted server verifies the trusted agent’s certificate and checks it against the “untrusted list” of compromised devices (step 3). <i>Id.</i> at 15:30-45.</p> <p>If the trusted agent can still be trusted, the trusted server will issue it a new digital certificate (step 3), which the trusted agent will check to ensure its validity (step 4). <i>Id.</i> at 15:45-57, 16:60-17:7, 17:16-23.</p>
<p>[1.j.] means for making a request for an authorization object required to be included within the repository for the apparatus to render the digital content; and</p>	<p>When the CTA is renewing a digital certificate or requesting an authentication message, it is both requesting and acquiring the digital certificate or authentication message, respectively, thereby satisfying both</p>

<p><u>This corresponds to the following algorithm:</u></p> <p>This limitation to contain the same first three steps as the “<i>means for authorizing</i>” limitation except the server repository invokes the transaction to “<i>request</i>” rather than “<i>acquire</i>” the authorization object.</p>	<p>limitations. <i>See</i> Ex. 1020 at 11:14-25, 6:38-55.</p>
<p>[1.k.] means for receiving the authorization object when it is determined that the request should be granted.</p> <p><u>This corresponds to the following algorithm:</u></p> <p>This limitation to contain two steps identical to steps 2 and 3 of the “<i>means for processing the request</i>” limitation.</p>	<p><u>First</u>, when, as part of the royalty checking protocol on the combined <u>Rosen/Hartrick</u> system, the MTA sends an authorization message to the CTA, if the transactions is interrupted or if a problem with the data is detected, the transaction will be aborted. <i>See</i> Ex. 1020 at 14:52-15:11 (describing aborting a transaction); Ex. 1021 at 16:26-33.</p> <p>As with other transactions, if the transaction is completed, the CTA will commit the transaction and send an acknowledgement to the MTA. <i>See id.</i> at 22:60-23:9; 16:52-58.</p> <p><u>Second</u>, in the context of <u>Rosen</u>’s credential ticket renewal, if the Identification Authority confirms it can renew the CTA’s credential, it will send the credential to the CTA, and upon successful transmission, the CTA will commit and will send an acknowledgement (step 4). <i>Id.</i> at 27:23-27, Fig. 28.</p> <p><u>Finally</u>, in the context of recertification</p>

	<p>of <u>Rosen</u>'s trusted agents, the trusted server will send and exchange information with the trusted agent, resulting in the final transmission of a new digital certificate. Ex. 1020 at 15:30-17:22 (describing message exchange during renewal protocol).</p> <p>Once the digital certificate is received, the CTA commits and sends a message indicating it has updated its certificate. <i>Id.</i> at 17:33-39.</p>
--	---

E232. Therefore, it is my opinion that Rosen in view of Hartrick renders claim 1 of the '576 patent obvious.

b. Claim 18 of the '576 Patent

E233. In the table below, I have mapped each claim element to the relevant paragraphs of my analysis.

<u>'576 Patent</u>	<u>Relevant Concepts and Paragraphs</u>
18. A method for controlling rendering of digital content on an apparatus having a rendering engine configured to render digital content and a storage for storing the digital content, said method comprising:	<p><u>“digital content”</u></p> <p>electronic objects. ¶¶ E21-E27;</p> <p><u>“apparatus”</u></p> <p>transaction devices. ¶¶ E11-E20</p> <p><u>“rendering engine”</u></p> <p>Transaction devices (e.g., computers) have applications for rendering content.</p>

	<p>¶¶ E11-E20.</p> <p><u>“storage”</u></p> <p>Transaction devices include storage for storing electronic objects. ¶¶ E11-E20</p>
[18.a.] specifying rights within said apparatus for digital content stored in said storage, said rights specifying how digital content can be rendered;	<p><u>“digital content stored in said storage”</u></p> <p>Transaction devices (<i>e.g.</i>, computers) can store electronic objects. ¶¶ E6-E20.</p> <p><u>“specifying rights ... specifying how digital content can be rendered”</u></p> <p>It would be obvious to store specialized SGML tags in the “Usage” field of the decryption ticket to specify whether content can be displayed, printed, copied, or transferred. ¶¶ E6-E10, E74-E98, E104-E114.</p>
[18.b.] storing digital content in said storage;	<p>storing digital content in storage in the transaction device. ¶¶ E6-E20</p>
[18.c.] receiving a request for rendering of said digital content stored in the storage;	<p><u>“request for rendering of said digital content stored in the storage”</u></p> <p>A customer uses a transaction application to send a request to the CTAs on the customer transaction device asking to access the decryption ticket and render content. ¶¶ E15-E20, E59, E99-E103, E115-E121, E142</p> <p>It would have been obvious to have authors specify usage information when depositing content with merchants, and</p>

	<p>to configure the MTA in a merchant transaction device to enforce specialized SGML tags stored in the “Usage” field of the decryption ticket. ¶¶ E99-E103, E115-E121</p> <p>It would have been obvious to configure MTAs to enforce the “Usage” field of the decryption ticket when selling content. ¶¶ E119-E121, E143.</p>
<p>[18.d.] checking whether said request is for a permitted rendering of said digital content in accordance with said rights specified within said apparatus;</p>	<p>It would have been obvious to configure MTAs to enforce the “Usage” field of the decryption ticket when selling content. ¶¶ E119-E121, E143.</p> <p>It would have been obvious to configure the CTA in a customer transaction device to enforce specialized SGML tags stored in the “Usage” field of the decryption ticket. ¶¶ E99-E103, E115-E121</p>
<p>[18.e.] processing the request to make said digital content available to the rendering engine for rendering when said request is for a permitted rendering of said digital content;</p>	<p>During the purchase transaction, after the CTA provides validation and payment information, and the MTA sends content to the CTA, the associated request for content is evaluated to determine whether it is for a permitted rendering. ¶¶ E38-E52 (transfers); E131-E134 (attachment of usage and content)</p> <p><u>“mak[ing] digital content available to the rendering engine”</u></p> <p>transmitting content from the MTA to</p>

<p>[18.f.] authorizing a repository for making the digital content available for rendering, wherein the digital content can be made available for rendering only by an authorized repository, the repository performing the steps of:</p>	<p>the CTA. ¶¶ E38-E52 (transfers);</p> <p><u>“repository”</u></p> <p>The customer transaction devices, each of which contains a trusted agent that is physically secure and that executes secure protocols for acquiring, storing, and rendering content. ¶¶ E2-E10; E24-E37, E136-E145</p> <p><u>“authorizing a repository”</u></p> <p>CTA must request and receive a payment credential ticket from an ATA through a ticket purchase transaction. ¶¶ E60, E66-E69</p> <p>It would have been obvious to incorporate the <u>Hartrick</u> authorization message into the <u>Rosen</u> system. ¶¶ E99-E103, E112-E114</p>
<p>[18.g.] making a request for an authorization object required to be included within the repository for rendering of the digital content; and</p>	<p><u>“making a request for an authorization object”</u></p> <p>CTA must request and receive a payment credential ticket from an ATA through a ticket purchase transaction. ¶¶ E60, E66-E69</p> <p>CTA must request and receive an authorization message to render content in a certain manner (<i>e.g.</i>, pay a fee to print). ¶¶ E99-E103, E112-E114</p>
<p>[18.h.] receiving the authorization object when it is determined that the</p>	<p>The ATA will evaluate whether a</p>

request should be granted.	request should be granted through a ticket purchase transaction, and if so, it will provide a credential ticket. ¶¶ E66-E69. MTA will determine whether an authorization message should be returned, and if so, it will send one to the CTA. ¶¶ E99-E103, E112-E114
----------------------------	--

E234. Therefore, it is my opinion that Rosen in view of Hartrick renders claim 18 of the '576 patent obvious.

c. Claims 2 and 19 of the '576 Patent

E235. Claims 2 and 19 depend from claims 1 and 18 and each add the following additional limitation to the claimed “apparatus for rendering digital content” and “method for controlling digital content,” respectively.

E236. Claim 2 specifies “the means for checking comprises means for comparing a requested use with a use specified by the rights.”

E237. Claim 19 specifies “the checking step comprises comparing a requested use with a use specified by the rights.”

E238. To enforce the specialized SGML tags stored in the “Usage” field (e.g., “print” or “copy-royalty” tags), the trusted agents would compare a requested use against the tags. Ex. 1021 at 3:28-4:9, 5:23-38. See ¶¶ E98-E103 (enforcement); E104-E122 (Rosen combination with Hartrick), *above*.

d. Claims 4 and 21 of the ‘576 Patent

E239. Claims 4 and 21 depend from claims 1 and 18 and each add the following additional limitation to the claimed “apparatus for rendering digital content” and “method for controlling digital content,” respectively.

E240. Claim 4 specifies “means for requesting a transfer of the digital content from an external memory to the storage.”

E241. Claim 21 specifies “requesting a transfer of the digital content from an external memory to the storage.”

E242. In the Rosen system, once a session has been established, a customer transaction device will request to purchase (*i.e.*, transfer) content, and after payment has been verified, the content stored at the merchant (“*external memory*”) will be packaged into a new electronic object and decryption ticket (“*digital content*”) and transmitted to the CTA and its associated memory. *Id.* at 19:13-39, 22:20-67, 23:23-40.

e. Claims 7 and 24 of the ‘576 Patent

E243. Claims 7 and 24 depend from claims 1 and 18 and each specifies “*wherein the digital content is video content.*”

E244. Rosen provides movies (“*video content*”) as an example of an electronic object. *Id.* at 4:55-56, 5:63-64.

f. Claims 8 and 25 of the ‘576 Patent

E245. Claims 8 and 25 depend from claims 1 and 18 and each specifies “wherein the apparatus is a portable device” and “wherein the method is implemented in a portable device,” respectively.

E246. Rosen shows that the system can be used in a portable device, such as an electronic wallet. *Id.* at 26:1-3.

g. Claims 9 and 26 of the ‘576 Patent

E247. Claims 9 and 26 depend from claims 1 and 18 and each specifies “wherein the rights are provided by a provider of the digital content.”

E248. In the Rosen/Hartrick system, an author (“*provider*”) can specify uses, conditions, and limitations on the author’s digital content. Ex. 1021 at 3:12-32, 9:11-14.

h. Claims 10 and 27 of the ‘576 Patent

E249. Claims 10 and 27 depend from claims 1 and 18 and each species “wherein the means for processing [step] comprises transmitting the digital content to the rendering machine.”

E250. In the *processing step*, a transaction application will transfer the electronic object to processors suitable to the “type of electronic object (e.g., movie, book, multimedia game, etc.).” Ex. 1020 at 8:22-34. If the electronic object is a movie, for example, it is rendered by playing it on a monitor for the user. *Id.* at 5:53-6:2.

i. Claims 15 and 32 of the ‘576 Patent

E251. Claims 15 and 32 depend from claims 1 and 18 and each specifies “wherein the rights are embodied in software instructions which implement the use privileges for the rights.”

E252. The rights disclosed in Hartrick, as combined with Rosen, are specified in the form of SGML tags that are enforced by software running in a trusted agent (“*embodied in software instructions*”), and the tags that specify how the electronic object can be used (“*use privileges*”).

j. Claim 29 of the ‘576 Patent

E253. Claim 29 depend from claim 18 and species “wherein the rendering engine is configured to render digital content into a medium that is not further protected by rights and the request is for rendering the digital content into a medium that is not further protected by rights.”

E254. Rosen shows that after security steps take place, the electronic object (“*digital content*”) is *rendered* in a way that is “*not further protected by the rights*.” Ex. 1020 at 23:46-51. When a movie is viewed, for example, the electronic object is converted into a viewable medium that is no longer under further rights restrictions but can be viewed by anyone within sight of the monitor. *Id.* at 5:53-6:2.

E255. Where content in an electronic object was a book or magazine, it would have been obvious to permit a customer print that content on a printer (perhaps for an additional fee), as shown in Hartrick.

k. Claim 34 of the ‘576 Patent

E256. Claim 34 recites the method of claim 18, “further comprises: requesting receipt of digital content stored externally; and receiving the digital content if it is permitted to receive the digital content.”

E257. In the Rosen system, once a session has been established, a customer transaction device will request to purchase (*i.e.*, transfer) content, and after payment has been verified, the content stored at the merchant (“*external memory*”) will be packaged into a new electronic object and decryption ticket (“*digital content*”) and transmitted to the CTA and its associated memory. *Id.* at 19:13-39, 22:20-67, 23:23-40.

IX. Conclusion

E258. For the reasons set forth above, it is my opinion that the subject matter recited in the claims of the Stefik patents was unpatentable as of November 1994.

List of Materials Considered

I have considered the Board's decisions to institute and final written decisions in the *inter partes* review proceedings filed by ZTE Corp. against ContentGuard in IPR2013-00133, -00134, -00137, and -00139. I also considered the exhibits listed in the chart below:

Exhibit #	Reference Name
1001	U.S. Patent No. 6,963,859
1002	File History of U.S. Patent No. 6,963,859
1003	U.S. Patent No. 7,523,072
1004	File History of U.S. Patent No. 7,523,072
1005	U.S. Patent No. 8,370,956
1006	File History of U.S. Patent No. 8,370,956
1007	U.S. Patent No. 8,393,007
1008	File History of U.S. Patent No. 8,393,007
1009	U.S. Patent No. 7,269,567
1010	File History of U.S. Patent No. 7,269,576
1011	U.S. Patent No. 7,225,160
1012	File History of U.S. Patent No. 7,225,160
1016	S. White & L. Comerford, ABYSS: A Trusted Architecture for Software Protection (1987)
1017	D. Denning, Cryptography and Data Security (1982)
1018	U.S. Patent No. 6,135,646 to Kahn

Appendix A: Materials Considered

Exhibit #	Reference Name
1019	Proceedings of the Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment, Journal of the Interactive Multimedia Association Intellectual Property Project (Jan. 1, 1994)
1020	U.S. Patent No. 5,557,518 to Rosen
1021	EP 0567800 A1 to Hartrick
1022	File History for U.S. Patent No. 6, 135,646 to Kahn
1023	U.S. Patent No. 5,629,980 to Stefik
1024	5,477,263 (O'Callegan)
1025	U.S. Patent No. 5,260,999, issued November 9, 1993 to Wyman
1026	J. Moffett et al., SPECIFYING DISCRETIONARY ACCESS CONTROL POLICY FOR DISTRIBUTED SYSTEMS, Computer Communications, vol 13 no 9, pp 571-580 (Nov. 1990)
1027	R. Mori & M. Kawahara, Superdistribution: The Concept and Architecture, The Transactions of The IEICE, Vol. E73, No. 7, pp. 1133-1146 (July 1990)
1028	European Patent Publication 0 268 139
1029	U.S. Patent No. 5,412,717 to Fischer
1030	The Copy-Protection Wars, PC Magazine
1031	US 4,658,093 to Hellman
1032	US 5,940,504 to Griswold
1033	G. Davida et al., Defending Systems Against Viruses through Cryptographic Authentication, IEEE 1989
1034	V. Cina et al., ABYSS: A Basic Yorktown Security System: PC Asset Protection Concepts, IBM Research Report Number RC 12401 (Dec. 1986)
1035	U.S. 5,109,413 to Comerford
1036	S. Weingart, Physical Security for the μ ABYSS System, Proceedings of the IEEE Symposium on Security and Privacy (Apr. 27-29, 1987)
1037	FIPS 140-1
1038	4,278,837 to Best
1039	William Aspray, The Stored Program Concept, IEEE Spectrum Sept.

Appendix A: Materials Considered

Exhibit #	Reference Name
	1990
1040	Glenn C. Reid, Thinking in Postscript, Addison-Wesley (1990)
1041	Bruce Schneier, Applied Cryptography (Chapters 1-3, 7, 12, 13, 17) (1994)
1042	Samuelson, Intellectual Property Rights for Digital Library and Hypertext Publishing Systems: An Analysis of Xanadu (Dec. 1991)
1043	Kahn & Cerf, "The Digital Library Project Volume 1: The World of Knowbots (Draft)" (1988)
1044	Internet Dreams - Stéfik foreword
1045	CNRI Digital Library Workshop
1046	Diffie-Hellman, New Directions in Crypto 1976
1047	X.509 Standard
1048	RFC 1422, Privacy Enhanced Mail (PEM)
1049	Steiner et al., Kerberos: An Authentication Service for Open Network Systems (MIT 1988)
1050	Needham & Schroeder, Using Encryption for Authentication in Large Networks of Computers (Dec. 1978)
1051	J. Saltzer & M. Schroeder, The Protection of Information in Computer Systems, Proceedings of the IEEE, Vol. 63, No. 9 (Sept. 1975)
1052	ContentGuard's Opening Claim Construction Brief, 2:13-cv-01112 (EDTX) Dkt No. 304
1053	U.S. Patent Application File History No. 09/778,001
1054	U.S. Patent Application File History Nos. 08/967,084 and 08/344,760
1055	U.S. Patent No. 4,977,594 to Shear
1056	Henry H. Perritt, Jr., " <i>Knowbots, Permissions Headers and Contract</i> ," Paper for the Conference on Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment (April 30, 1993)
1057	<i>Dyad: A System for Using Physically Secure Coprocessors</i> , J.D. Tygar and Bennett Lee, Proceedings of the Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment, Journal of the Interactive Multimedia Association Intellectual Property Project (Jan. 1, 1994)
1058	" <i>Copyright and Information Services in the Context of the National</i>

Appendix A: Materials Considered

Exhibit #	Reference Name
	<i>Research and Education Network</i> ,” R.J. Linn, Proceedings of the Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment, Journal of the Interactive Multimedia Association Intellectual Property Project (Jan. 1, 1994)
1059	“ <i>The Strategic Environment for Protecting Multimedia</i> ,” Brian Kahin, Proceedings of the Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment, Journal of the Interactive Multimedia Association Intellectual Property Project (Jan. 1, 1994)
1060	Amir Herzberg, Shlomit S. Pinter, “Public Protection of Software,” in <i>Advances in Cryptology: Proc. Crypto 85</i> , Hugh C. Williams (cd.), pp. 158 (1986)
1061	S. White & L. Comerford, <i>ABYSS: A Trusted Architecture for Software Protection</i> (1990)
1062	T. Berners-Lee & D. Connolly, RFC 1866, <i>Hypertext Markup Language - 2.0</i> (Nov. 1995)