

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,
Petitioner,

v.

SECURITY FIRST INNOVATIONS, LLC
Patent Owner.

Case No. IPR2024-00215
Patent No. 10,452,854

DECLARATION OF SAMRAT BHATTACHARJEE, PH.D.

TABLE OF CONTENTS

I.	PERSONAL AND PROFESSIONAL BACKGROUND	1
II.	MATERIALS REVIEWED AND CONSIDERED	4
III.	MY UNDERSTANDING OF PATENT LAW	7
	A. Anticipation	9
	B. Obviousness.....	9
IV.	UNPATENTABILITY GROUNDS.....	12
V.	THE '854 PATENT	15
	A. Specification	15
	B. Person of Ordinary Skill in the Art (“POSA”).....	17
	C. Prosecution History	20
	D. Challenged Claims.....	20
VI.	CLAIM INTERPRETATION	23
VII.	GROUND 1: CLAIMS 1-10 ARE ANTICIPATED OR ALTERNATIVELY OBVIOUS OVER ORSINI.....	24
	A. Orsini (EX1005)	24
	B. Claim 1:	35
	1. 1[PRE] A method for securely storing a data set, the method comprising:.....	35
	2. [1A] receiving an external key from an external storage system,.....	36
	a. Orsini discloses [1A] (“ <i>receiving an external key from an external storage system</i> ”).....	37
	i. Encryption keys received from and stored at Orsini’s “external” key management.....	38
	ii. Public-key encryption keys received from a certificate authority	44
	b. Alternatively, [1A] would have been obvious over Orsini.....	49
	3. [1B] generating a plurality of data chunks based on the data set, such that the data set can be reconstructed using at least a minimum number of the plurality of data chunks:.....	60

4.	[1B1] wherein generating the data chunks comprises: distributing the data set into a plurality of shares, wherein each of the shares comprises less than all of the data set, ...	63
5.	[1B2] [wherein generating the data chunks comprises:] accessing a plurality of distinct encryption keys, ...	64
6.	[1B3] [wherein generating the data chunks comprises:] encrypting each of the shares with a respective one of the plurality of distinct encryption keys.....	64
7.	[1C] performing an encryption operation based on the external key to further secure the plurality of data chunks; and	65
a.	Orsini discloses [1C] (“ <i>performing an encryption operation based on the external key to further secure the plurality of data chunks</i> ”)	66
i.	Orsini discloses performing encryption steps “in different order”	66
ii.	Orsini discloses repeating encryption steps “multiple times.”	69
iii.	Encrypting With a Public Key	73
b.	POSAs Had Reason To “ <i>Perform[] an Encryption Operation Based on the External Key to Further Secure the Plurality of Data Chunks</i> ”.....	75
i.	POSAs had reason to re-encrypt data chunks with an external key	75
ii.	POSAs had reason to re-encrypt data chunks with an externally-stored group key.	78
iii.	POSAs had reason to reorder data within each chunk based on an external key.....	80
iv.	POSAs had reason to encrypt [1B3]’s keys with an externally-stored key-encryption key.	83
8.	[1D] storing with the plurality of data chunks data indicative of at least one of the distinct encryption keys on a plurality of different storage devices.....	86
C.	Claim 2: The method of claim 1, wherein storing with the plurality of data chunks data indicative of at least one of the distinct	

encryption keys comprises storing each of the plurality of data chunks with at least one of the distinct encryption keys.	91
D. Claim 3: The method of claim 1, wherein storing each of the plurality of data chunks with at least one of the distinct encryption keys comprises including with a particular data chunk of the plurality of data chunks data indicative of an encryption key that was used to encrypt a different data chunk of the plurality of data chunks.	91
E. Claim 4: The method of claim 1, wherein distributing the data set into a plurality of shares comprises using a random technique or pseudorandom technique.	93
F. Claim 5: The method of claim 1, wherein distributing the data set into a plurality of shares comprises using a deterministic technique or pseudorandom technique.	95
G. Claim 6: The method of claim 1, wherein each share of the plurality of shares comprises a substantially random distribution of a subset of the data set.	96
H. Claim 7: The method of claim 1, wherein performing an encryption operation based on the external key to further secure the plurality of data chunks comprises encrypting primary data with the external key.	96
I. Claim 8: The method of claim 1, wherein distributing the data set into a plurality of shares comprises: generating a split encryption key based on the external key; and distributing the data set based on the split encryption key.	97
J. Claim 9: The method of claim 8, further comprising encrypting each of the shares with the external key.	100
K. Claim 10: The method of claim 9, wherein the data set is recoverable using at least a minimum number of the plurality of data chunks and the external key.	101
VIII. GROUND 2: Claims 1-10 Would Have Been Obvious Over FOSTER- HAYHURST.	103
A. Foster (EX1006)	103
B. Hayhurst (EX1007).	109
C. The Foster-Hayhurst Combination	112

D. Claim 1:	120
1. 1[PRE] A method for securely storing a data set, the method comprising:	120
2. [1A] receiving an external key from an external storage system,.....	121
3. [1B] generating a plurality of data chunks based on the data set, such that the data set can be reconstructed using at least a minimum number of the plurality of data chunks:.....	126
4. [1B1] wherein generating the data chunks comprises: distributing the data set into a plurality of shares, wherein each of the shares comprises less than all of the data set, ...	131
5. [1B2] [wherein generating the data chunks comprises:] accessing a plurality of distinct encryption keys, ...	133
6. [1B3] [wherein generating the data chunks comprises:] encrypting each of the shares with a respective one of the plurality of distinct encryption keys,.....	134
7. [1C] performing an encryption operation based on the external key to further secure the plurality of data chunks; and	135
8. [1D] storing with the plurality of data chunks data indicative of at least one of the distinct encryption keys on a plurality of different storage devices.....	138
E. Claim 2: The method of claim 1, wherein storing with the plurality of data chunks data indicative of at least one of the distinct encryption keys comprises storing each of the plurality of data chunks with at least one of the distinct encryption keys.	142
F. Claim 3: The method of claim 1, wherein storing each of the plurality of data chunks with at least one of the distinct encryption keys comprises including with a particular data chunk of the plurality of data chunks data indicative of an encryption key that was used to encrypt a different data chunk of the plurality of data chunks.	143
G. Claim 4: The method of claim 1, wherein distributing the data set into a plurality of shares comprises using a random technique or pseudorandom technique.	144

H. Claim 5: The method of claim 1, wherein distributing the data set into a plurality of shares comprises using a deterministic technique or pseudorandom technique.....	149
I. Claim 6: The method of claim 1, wherein each share of the plurality of shares comprises a substantially random distribution of a subset of the data set.	149
J. Claim 7: The method of claim 1, wherein performing an encryption operation based on the external key to further secure the plurality of data chunks comprises encrypting primary data with the external key.....	150
K. Claim 8: The method of claim 1, wherein distributing the data set into a plurality of shares comprises: generating a split encryption key based on the external key; and distributing the data set based on the split encryption key.	151
L. Claim 9: The method of claim 8, further comprising encrypting each of the shares with the external key.....	154
M. Claim 10: The method of claim 9, wherein the data set is recoverable using at least a minimum number of the plurality of data chunks and the external key.....	155
IX. APPENDIX: CLAIM LISTING.....	1

APPENDIX LISTING OF EXHIBITS

Exhibit	Description
1001	U.S. Patent No. 10,452,854 (“’854 patent”)
1002	Prosecution History of U.S. Patent No. 10,452,854
1003	Declaration of Samrat Bhattacharjee, Ph.D.
1004	Curriculum Vitae of Samrat Bhattacharjee, Ph.D.
1005	U.S. Patent Application Publication 2004/0049687A1 (“Orsini”)
1006	U.S. Patent Application Publication 2003/0200176A1 (“Foster”)
1007	U.S. Patent Application Publication 2004/0216164A1 (“Hayhurst”)
1009	Security First Innovation’s 2023-08-14 Infringement Contentions, Chart A in <i>Security First Innovations, LLC v. Google LLC</i> , 2-23-cv-00097 (E.D. Va.)
1010	Redline Comparison of U.S. Patent Application Publication 2004/0049687A1 to U.S. Patent No. 10,452,854
1011	U.S. Patent No. 5,991,414
1012	U.S. Patent Application Publication No. 2002/0169842
1013	U.S. Patent No. 6,393,568
1014	U.S. Patent No. 6,868,160
1015	U.S. Patent Application Publication No. 2006/0053112
1016	U.S. Patent Application Publication No. 2004/0005061
1017	U.S. Patent No. 5,412,722
1018	U.S. Patent No. 7,689,829
1019	U.S. Patent Application Publication No. 2003/0081784
1020	U.S. Patent No. 7,634,650
1021	U.S. Patent Application Publication No. 2004/0143733
1022	U.S. Patent Application Publication No. 2002/0126850
1023	U.S. Patent No. 5,812,671
1024	U.S. Patent Application Publication No. 2003/0012386
1025	U.S. Patent No. 5,963,646
1026	U.S. Patent No. 5,416,841
1027	U.S. Patent No. 7,251,828
1028	U.S. Patent Application Publication No. 2003/0167314
1029	U.S. Patent No. 6,598,161
1030	U.S. Patent Application Publication No. 2005/0198490
1031	U.S. Patent Application Publication No. 2002/0169808
1032	U.S. Patent No. 6,345,101
1033	U.S. Patent No. 8,707,034
1034	U.S. Patent Application Publication No. 2003/0105719

1035	U.S. Patent No. 4,238,853
1036	U.S. Patent No. 6,819,766
1037	U.S. Patent Application Publication No. 2004/0172476
1039	U.S. Patent No. 7,885,405
1040	U.S. Patent Application Publication No. 2005/0210269
1041	U.S. Patent Application Publication No. 2005/0120065
1043	U.S. Patent Application Publication No. 2001/0037474
1044	U.S. Patent Application Publication No. 2003/0191938
1045	U.S. Patent Application Publication No. 2003/0028493
1046	U.S. Patent Application Publication No. 2003/0016820
1047	U.S. Patent Application Publication No. 2002/0042859
1048	U.S. Patent Application Publication No. 2003/0188153
1049	PCT Application Publication No. WO 2004/057461
1050	U.S. Patent No. 7,529,834
1051	U.S. Patent Application Publication No. 2006/0034457
1052	U.S. Patent Application Publication No. 2005/0256972
1053	U.S. Patent Application Publication No. 2004/0049700
1054	U.S. Patent No. 5,675,653
1055	U.S. Patent Application Publication No. 2006/0062384
1056	U.S. Patent No. 6,490,353
1061	U.S. Patent Application Publication No. 2006/0059347
1062	U.S. Patent Application Publication No. 2006/0064386
1064	Google LLC’s Proposed Claim Terms for Construction in <i>Security First Innovations, LLC v. Google, LLC</i> , 2:23-cv-00097-JWK-RJK (Nov. 11, 2023)
1065	Letter from A. Mayron to D. Perlson re Preliminary Proposed Constructions in <i>Security First Innovations, LLC v. Google, LLC</i> , 2:23-cv-00097-JWK-RJK (Nov. 11, 2023)
1066	U.S. Patent Application Publication No. 2004/0228479
1067	U.S. Patent Application Publication No. 2004/0039925

I, Samrat Bhattacharjee, Ph.D., declare:

1. I have been retained by Wolf, Greenfield & Sacks, P.C., counsel for Petitioner Google LLC, to assess claims 1-10 (the “challenged claims”) of U.S. Patent No. 10,452,854 (“the ’854 patent”) (EX1001). I am being compensated for my time at my standard rate of \$700.00 per hour, plus actual expenses. My compensation is not dependent in any way upon the outcome of the *inter partes* review of the ’854 patent.

I. PERSONAL AND PROFESSIONAL BACKGROUND

2. My qualifications for forming the opinions set forth in this declaration are summarized below and explained in more detail in my current curriculum vitae, provided as EX1004. EX1004 also includes a list of my publications and patents.

3. I am a tenured professor in the Computer Science Department at the University of Maryland. I am also an Affiliate Professor with the Department of Electrical and Computer Engineering at the University of Maryland. I have been an appointed faculty member at the University of Maryland since 1999.

4. I received a Ph.D. in computer science from the Georgia Institute of Technology in 1999. Before that I received a B.S. degree in Mathematics and Computer Science with Highest Distinction (Summa Cum Laude) and was recognized as the Outstanding Department Major at Georgia College and State

University, Milledgeville, Georgia in 1994.

5. I have been involved in the research and development of computing systems for over two decades. I perform research and I teach in the areas of computer networks, distributed systems, computer security, cryptography, and operating systems, among others. A major theme of my research has been the development of technology to improve the performance and security of data transfers and data storage on networks like the Internet. My research has examined problems ranging from network and operating system support audio and video streaming, large scale Internet video distribution, the design of novel network architectures for better and secure content distribution and storage, efficiency of content distribution over wireless networks, and so on.

6. For example, starting in the late 1990s, the focus of my research was the development of network and operating system technology to design a new network architecture that allowed for more efficient data transfer. Part of my Ph.D. research was to develop new and more efficient architectures for video delivery on the Internet, and I have published papers on this architecture during my graduate studies. I continued to work on video delivery and security and storage of digital content as a faculty member, and have published various papers in these areas. During 2007, I was a visiting researcher at AT&T Labs, and one of the projects I focused on was a video content delivery platform. This work resulted in both

publications and a granted U.S. patent (U.S. 8,752,100 B2). Among other research, I have authored papers and developed research software for public-key infrastructure KeyChains and a cryptographic library for “chit”-based security.

7. Along with working on research problems related to computer networking, computer security, cryptography, and video distribution, I routinely taught this material in both graduate and undergraduate courses. A number of my papers in these domains are with students at the University of Maryland, many of whom started in these areas after taking my classes.

8. For over 25 years, I have developed professional and academic experience in the field of cryptographic techniques for computer systems. For example, I have authored or co-authored over 100 articles in peer-reviewed journals, conference proceedings, and workshops relating to computer networking, computer system, and data security. Over the years I have received several fellowships, prizes, and awards for teaching excellence and technical papers, including an Alfred P. Sloan Jr. Fellowship in 2004 and the National Science Foundation CAREER award in 2001. I have also received an award from the SIGCOMM foundation for a “Test of Time” paper (that is given to an influential paper published ten years ago). My research work has been supported by multiple grants from the US National Science Foundation, and the Department of Defense. I have also started a Joint Ph.D. program with the University of Maryland and the

Max Planck Society in Germany, and co-founded the annual Cornell, Maryland, Max Planck Research School that provides research exposure to about 80 students from across the world during a week-long school.

9. I have served on numerous paper and proposal review panels and participated on program committees for ACM/IEEE groups, technical journals, and conferences in Networking, Security and Systems in the aforementioned areas of computer science. I have created research software for all of my projects, much of which is available online for others to build on.

10. I am a named inventor on four U.S. Patents. These patents are generally related to computer networking and the delivery of services over networks and video delivery systems.

11. I have served as an expert witness and technical consultant in litigation and Inter Partes Review matters concerning videoconferencing, data conferencing and screen sharing, voice over IP (VoIP) telephony, multimedia networking, distributed systems, operating systems, computer networks, and datacenter networking, among others. I have testified in several trials and depositions as an expert witness.

II. MATERIALS REVIEWED AND CONSIDERED

12. My findings, as explained below, are based on my years of education, research, experience, and background in the field of cryptographic techniques for

computer systems, as well as my investigation and study of relevant materials for this declaration. When developing the opinions set forth in this declaration, I assumed the perspective of a person having ordinary skill in the art, as set forth in Section V.A below. In forming my opinions, I have studied and considered the materials identified in the list below.

Exhibit	Description
1001	U.S. Patent No. 10,452,854 (“’854 patent”)
1002	Prosecution History of U.S. Patent No. 10,452,854
1005	U.S. Patent Application Publication 2004/0049687A1 (“Orsini”)
1006	U.S. Patent Application Publication 2003/0200176A1 (“Foster”)
1007	U.S. Patent Application Publication 2004/0216164A1 (“Hayhurst”)
1009	Security First Innovation’s 2023-08-14 Infringement Contentions, Chart A in <i>Security First Innovations, LLC v. Google LLC</i> , 2-23-cv-00097 (E.D. Va. Aug. 14, 2023)
1010	Redline Comparison of U.S. Patent Application Publication 2004/0049687A1 to U.S. Patent No. 10,452,854
1011	U.S. Patent No. 5,991,414
1012	U.S. Patent Application Publication No. 2002/0169842
1013	U.S. Patent No. 6,393,568
1014	U.S. Patent No. 6,868,160
1015	U.S. Patent Application Publication No. 2006/0053112
1016	U.S. Patent Application Publication No. 2004/0005061
1017	U.S. Patent No. 5,412,722
1018	U.S. Patent No. 7,689,829
1019	U.S. Patent Application Publication No. 2003/0081784
1020	U.S. Patent No. 7,634,650
1021	U.S. Patent Application Publication No. 2004/0143733
1022	U.S. Patent Application Publication No. 2002/0126850
1023	U.S. Patent No. 5,812,671
1024	U.S. Patent Application Publication No. 2003/0012386
1025	U.S. Patent No. 5,963,646
1026	U.S. Patent No. 5,416,841
1027	U.S. Patent No. 7,251,828
1028	U.S. Patent Application Publication No. 2003/0167314

1029	U.S. Patent No. 6,598,161
1030	U.S. Patent Application Publication No. 2005/0198490
1031	U.S. Patent Application Publication No. 2002/0169808
1032	U.S. Patent No. 6,345,101
1033	U.S. Patent No. 8,707,034
1034	U.S. Patent Application Publication No. 2003/0105719
1035	U.S. Patent No. 4,238,853
1036	U.S. Patent No. 6,819,766
1037	U.S. Patent Application Publication No. 2004/0172476
1039	U.S. Patent No. 7,885,405
1040	U.S. Patent Application Publication No. 2005/0210269
1041	U.S. Patent Application Publication No. 2005/0120065
1043	U.S. Patent Application Publication No. 2001/0037474
1044	U.S. Patent Application Publication No. 2003/0191938
1045	U.S. Patent Application Publication No. 2003/0028493
1046	U.S. Patent Application Publication No. 2003/0016820
1047	U.S. Patent Application Publication No. 2002/0042859
1048	U.S. Patent Application Publication No. 2003/0188153
1049	International Application Publication No. WO 2004/057461
1050	U.S. Patent No. 7,529,834
1051	U.S. Patent Application Publication No. 2006/0034457
1052	U.S. Patent Application Publication No. 2005/0256972
1053	U.S. Patent Application Publication No. 2004/0049700
1054	U.S. Patent No. 5,675,653
1055	U.S. Patent Application Publication No. 2006/0062384
1056	U.S. Patent No. 6,490,353
1061	U.S. Patent Application Publication No. 2006/0059347
1062	U.S. Patent Application Publication No. 2006/0064386
1064	Google LLC's Proposed Claim Terms for Construction in <i>Security First Innovations, LLC v. Google, LLC</i> , 2:23-cv-00097-JWK-RJK (Nov. 11, 2023)
1065	Letter from A. Mayron to D. Perlson re Preliminary Proposed Constructions in <i>Security First Innovations, LLC v. Google, LLC</i> , 2:23-cv-00097-JWK-RJK (Nov. 11, 2023)
1066	U.S. Patent Application Publication No. 2004/0228479
1067	U.S. Patent Application Publication No. 2004/0039925

III. MY UNDERSTANDING OF PATENT LAW

13. In developing my opinions, I discussed various relevant legal principles with Petitioner's attorneys. I understood these principles when they were explained to me and have relied upon such legal principles, as explained to me, in the course of forming the opinions set forth in this declaration. My understanding in this respect is as follows:

14. I understand that "*inter partes* review" (IPR) is a proceeding before the United States Patent & Trademark Office for evaluating the patentability of an issued patent's claims based on prior-art patents and printed publications.

15. I understand that, in this proceeding, Petitioner has the burden of proving that the challenged claims of the '854 patent are unpatentable by a preponderance of the evidence. I understand that "preponderance of the evidence" means that a fact or conclusion is more likely true than not true.

16. I understand that, in IPR proceedings, claim terms in a patent are given their ordinary and customary meaning as understood by a person of ordinary skill in the art ("POSA") in the context of the entire patent and the prosecution history pertaining to the patent. If the specification or prosecution history provides a special definition for a claim term that differs from the meaning the term would otherwise possess, that special definition controls. If a claim element is expressed as a "means" for performing a specified function, I understand that it covers the

corresponding structure described in the specification and equivalents of the described structure. I have applied these standards in preparing the opinions in this declaration.

17. I understand that determining whether a particular patent or printed publication constitutes prior art to a challenged patent claim can require determining the effective filing date (also known as the priority date) to which the challenged claim is entitled. I understand that for a patent claim to be entitled to the benefit of the filing date of an earlier application to which the patent claims priority, the earlier application must have described the claimed invention in sufficient detail to convey with reasonable clarity to the POSA that the inventor had possession of the claimed invention as of the earlier application's filing date. I understand that a disclosure that merely renders the claimed invention obvious is not sufficient written description for the claim to be entitled to the benefit of the filing date of the application containing that disclosure.

18. I understand that for an invention claimed in a patent to be patentable, it must be, among other things, new (novel—*i.e.*, not anticipated) and not obvious from the prior art. My understanding of these two legal standards is set forth below.

A. Anticipation

19. I understand that, for a patent claim to be “anticipated” by the prior art (and therefore not novel), each and every limitation of the claim must be found, expressly or inherently, in a single prior-art reference. I understand that a claim limitation is disclosed for the purpose of anticipation if a POSA would have understood the reference to disclose the limitation based on inferences that a POSA would reasonably be expected to draw from the explicit teachings in the reference when read in light of the POSA’s knowledge and experience.

20. I understand that a claim limitation is inherent in a prior art reference if that limitation is necessarily present when practicing the teachings of the reference, regardless of whether a person of ordinary skill recognized the presence of that limitation in the prior art.

B. Obviousness

21. I understand that a patent claim may be unpatentable if it would have been obvious in view of a single prior-art reference or a combination of prior-art references.

22. I understand that a patent claim is obvious if the differences between the subject matter of the claim and the prior art are such that the subject matter as a whole would have been obvious to a person of ordinary skill in the relevant field at

the time the invention was made. Specifically, I understand that the obviousness question involves a consideration of:

- 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
- if present, objective factors indicative of non-obviousness, sometimes referred to as “secondary considerations.” To my knowledge, the Patent Owner has not asserted any such secondary considerations with respect to the ’854 patent.

23. I understand that in order for a claimed invention to be considered obvious, a POSA must have had a reason for combining teachings from multiple prior-art references (or for altering a single prior-art reference, in the case of obviousness in view of a single reference) in the fashion proposed.

24. I further understand that in determining whether a prior-art reference would have been combined with other prior art or with other information within the knowledge of a POSA, the following are examples of approaches and rationales that may be considered:

- combining prior-art elements according to known methods to yield predictable results;

- simple substitution of one known element for another to obtain predictable results;
- use of a known technique to improve similar devices in the same way;
- applying a known technique to a known device ready for improvement to yield predictable results;
- applying a technique or approach that would have been “obvious to try,” *i.e.*, choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success.
- known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to one of ordinary skill in the art;
- some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior-art reference or to combine prior-art reference teachings to arrive at the claimed invention. I understand that this teaching, suggestion or motivation may come from a prior-art reference or from the knowledge or common sense of one of ordinary skill in the art.

25. I understand that a universal motivation known in a particular field to improve technology can provide a motivation to combine prior art references even

absent any hint or suggestion in the references themselves. I also understand that obviousness is determined in light of all the facts and that a given course of action often has simultaneous advantages and disadvantages, and that this does not necessarily obvious motivation to combine teachings from multiple references.

26. I understand that for a single reference or a combination of references to render the claimed invention obvious, a POSA must have been able to arrive at the claimed invention by modifying, implementing, or combining the teachings of the applied references.

IV. UNPATENTABILITY GROUNDS

27. The table below identifies the references, applicable claims, and basis for each ground of unpatentability.

Ground Number and Reference(s)		Claims	Basis
1	Orsini (EX1005)	1-10	Anticipation and/or Obviousness
2	Foster (EX1006) and Hayhurst (EX1007)	1-10	Obviousness

28. I understand that the above-identified references are prior art (at least) based on the following publication dates, each of which is more than one year before the earliest-alleged priority date (2005-11-18):

Reference	Publication Date
Orsini	2004-03-11 (EX1005, code (43))
Foster	2003-10-23 (EX1006, code (43))
Hayhurst	2004-10-28 (EX1007, code (43))

29. As I discuss below (§V), the '854 patent discusses methods that “securely stor[e]” a data set by “split[ting] the data into multiple portions” and “encrypt[ing]” the data, either before or after it has been split (or both). EX1001, Abstract; *see also id.* 2:45-54. The encrypted portions of data can then be stored in different locations. EX1001, 2:56-58.

30. To perform encryption, the '854 patent relies on prior-art “encryption methods or algorithms” that were published and well-known to POSAs years before the patent’s earliest alleged November 2005 priority date. EX1001, 1:18-19, 61:5-46 (disclosing patent relies on encryption “algorithm[s that are] well known in the art” and “commercially available”). As I discuss below (§V), the '854 patent likewise relies on conventional and well-known techniques for splitting data into multiple portions. The purported “novelty” of the challenged claims thus is not directed to either of those concepts. Beyond conventional encryption and data-splitting operations, the challenged method claims also recite the use of an “external server” that stores an “external key” that the system uses to perform a post-encryption step, *i.e.*, to encrypt the data ***again*** (*i.e.*, a second encryption operation) after the data has already been split and encrypted a first time. But re-encrypting data and using an external key to do so were also conventional techniques at the time of the purported invention. The challenged claims are either

not novel and/or obvious to persons of skill in the art (“POSAs”), for the reasons I discuss below (§§VII-VIII).

31. Ground 1 (§VII below) demonstrates that Orsini (EX1005) anticipates and/or renders obvious claims 1-10. Orsini is Patent Owner’s own patent application that published more than a year before the earliest purported priority date for the ’854 patent. Like the ’854 patent, Orsini discloses securely storing a data set by splitting it into multiple portions, encrypting those portions, and then storing the encrypted portions in different locations. *See* §VII.A below. Orsini also explicitly teaches—and even has claims reciting—performing a post-encryption step on the encrypted data portions before they are stored. And Orsini teaches performing encryption using external keys received from external servers. In my opinion, the ’854 patent attempts to claim the same technology (or obvious variants thereof) that Patent Owner disclosed years earlier in Orsini, for the reasons I discuss below (§VII).

32. Ground 2 (§VIII below) demonstrates that claims 1--10 are also unpatentable over Foster (EX1006) in combination with Hayhurst (EX1007). Foster secures a data set by splitting it into multiple portions, encrypting those portions with individual encryption keys, encrypting those keys with a common encryption key received from an external storage system, and transmitting the encrypted portions and keys for subsequent storage. *See* §VIII.A below. Hayhurst

discloses, among other things, techniques for storing encrypted portions of data—like those Foster generates—on a plurality of storage devices. *See* §VIII.B below. As I discuss below (§VIII.C) POSAs would have had reason to combine Foster’s encryption technique with Hayhurst’s storage technique resulting in a system that performs the methods recited in claims 1-10.

V. THE ’854 PATENT

A. Specification

33. The ’854 patent relates to techniques for securing electronic data from unauthorized access. ’854 patent, 1:25-26. As the ’854 states and POSAs were aware, by 2005, individuals and businesses were storing and transmitting an ever-increasing amount of sensitive information over computer networks (*e.g.*, the Internet and corporate networks). ’854 patent, 1:30-37. According to the ’854 patent, prior-art techniques for securing such information included cryptography, which “refers to protecting data by transforming, or encrypting, it into an unreadable format” so “[o]nly those who possess the key(s) to the encryption can decrypt the data into a useable format.” ’854 patent, 1:47-51. The ’854 patent alleges that then-existing cryptographic techniques were problematically “reliant on the user” to keep secure the encryption keys used to secure (encrypt) and access (decrypt) data. ’854 patent, 1:60-62. This was allegedly problematic because,

according to the '854 patent, users often stored keys in unsecure systems or lost access to devices storing their keys (*e.g.*, mobile phones). '854 patent, 1:63-2:13.

34. To purportedly overcome this alleged problem, the '854 patent discusses a system for securing a data set that is “user-independent.” '854 patent, 2:42-44. The system is user-independent because it relies on “one or more secure servers” that a user can access but from which “actual [encryption] keys and other authentication data” is not released. '854 patent, 8:66-9:6. In other words, the encryption keys remain secured on the cryptographic servers and are not given to the user. *Id.* According to the '854 patent, “[t]his server-centric storage of keys...provides for user-independent security, portability, availability, and straightforwardness.” '854 patent, 9:6-8.

35. The '854 patent's cryptographic server(s) perform an encryption method that involves three basic steps: [1] “parsing, splitting, and/or separating” a data set “into two or more parts or portions,” [2] encrypting the data set (either before or after) it has been parsed, split, and/or separated, and [3] storing the “parsed, split, and/or separated data that has been encrypted in one location or in multiple locations.” '854 patent, 2:45-58. Of these three steps, the '854 patent says that steps [1] and [2] can be “repeated” such that the data set is parsed, split, and/or separated and/or encrypted multiple times. '854 patent, 2:52-55.

36. As I discuss below in §V.D, the challenged method claims recite performing the '854 patent's three basic encryption steps and further recite the use of an "external [encryption] key" to perform an additional "encryption operation" to further secure the "data chunks" of the data set after the data set has already been parsed and encrypted. The patent describes an external key as one that is "provided by a user or an external system." '854 patent, 77:59-61. This "external key" can be used to "post encrypt" the data set after it has first been parsed and encrypted. '854 patent, 79:15-19; *see also* FIG. 36 (step 3619). The '854 patent ascribes no advantages to using external keys compared to non-external keys or to receiving such keys from external storage systems compared to non-external storage systems.

B. Person of Ordinary Skill in the Art ("POSA")

37. I have been informed and understand that for purposes of assessing whether prior-art references disclose every element of a patent claim (thus "anticipating" the claim) and/or would have rendered the claim obvious, the patent and the prior-art references must be assessed from the perspective of a person having ordinary skill in the art ("POSA") to which the patent is related, based on the understanding of that person at the time of the patent claim's priority date. I have been informed and understand that a POSA is presumed to be aware of all

pertinent prior art and the conventional wisdom in the art, and is a person of ordinary creativity. I have applied this standard throughout my declaration.

38. The '854 patent involves technology in the field of securing data from unauthorized access or use. *See* EX1001, 1:25-26 (“The present invention relates in general to a system for securing data from unauthorized access or use.”). I have been asked to provide my opinions as to the state of the art in this field by 2005. I use this timeframe because the face of the '854 patent indicates an earliest claimed priority date of November 18, 2005. Whenever I offer an opinion in this declaration about the knowledge of a POSA, the manner in which a POSA would have understood the claims of the '854 patent or its description, the manner in which a POSA would have understood the prior art, or what a POSA would have been led to do based on the prior art, I am referencing the 2005 timeframe, even if I do not say so specifically in each case. Demonstrating that the '854 patent's claims were anticipated or would have been obvious by November 2005 demonstrates that they remained so any time after (including as of the November 20, 2006, filing date of the non-provisional application corresponding to U.S. Patent No. 8,009,830, which the '854 patent claims priority to), because the POSA's knowledge only increases over time.

39. I understand that the Patent Owner may attempt to prove that the alleged invention recited in the challenged claims was conceived at some time

prior to the earliest claimed priority date on the face of the patent. At the time of this declaration, I am unaware of the Patent Owner having alleged any earlier conception date or produced any evidence to establish any earlier conception date.

40. In my opinion, the person of ordinary skill in the art (POSA) at the time of the '854 patent's November 18, 2005 earliest claimed priority date would have at least a bachelor's degree in computer science, computer engineering, or a related field, with three (3) years of experience in the area of securing data from unauthorized use or access. A higher level of education may substitute for less experience. This person would have been capable of understanding and applying the teachings of the '854 patent and the prior-art references discussed in this declaration.

41. By 2005, I held a Ph.D. in computer science from the Georgia Institute of Technology, and I had over 10 years of experience with securing data from unauthorized use or access. *See* §I above. Therefore, I was a person of more than ordinary skill in the art during the relevant timeframe. However, I worked with many people who fit the characteristics of the POSA, and I am familiar with their level of skill. When developing the opinions set forth in this declaration, I assumed the perspective of a person having ordinary skill in the art, as set forth above.

C. Prosecution History

42. The claims were allowed without any rejections by the examiner. EX1002, 33-38. I note that Orsini issued as U.S. Patent No. 7,391,865, which was cited in an Information Disclosure Statement (IDS) by the application. EX1002, 66. However, neither Orsini nor the '865 patent were substantively discussed during prosecution of the '854 patent. I also note that a different patent sharing some figures with Orsini, U.S. Patent No. 8,644,502, was noted by the examiner in the Notice of Allowance (EX1002, 39), but that patent was also not substantively discussed during prosecution of the '854 patent. Neither Hayhurt nor Foster (nor any patent or patent application related thereto) was cited or discussed during prosecution of the '854 patent.

D. Challenged Claims

43. Claims 1-10 are reproduced in the Claim Listing (§0 below), and independent claim 1 is reproduced below with limitations labeled 1[PRE]-[1D]:

1[PRE] A method for securely storing a data set, the method comprising:

[1A] receiving an external key from an external storage system,

[1B] generating a plurality of data chunks¹ based on the data set, such that the data set can be reconstructed using at least a minimum number of

¹ I note that the claims of the '854 patent and Patent Owner's U.S. Patent No.

11,068,609 (the "'609 patent") (challenged in IPR2024-00214) both recite the term

the plurality of data chunks, wherein generating the data chunks comprises:

[1B1] distributing the data set into a plurality of shares, wherein each of the shares comprises less than all of the data set,

[1B2] accessing a plurality of distinct encryption keys,

[1B3] encrypting each of the shares with a respective one of the plurality of distinct encryption keys,

[1C] performing an encryption operation based on the external key to further secure the plurality of data chunks; and

[1D] storing with the plurality of data chunks data indicative of at least one of the distinct encryption keys on a plurality of different storage devices.²

“*data chunks*,” but the term is used differently in each claim set. The ’609 patent generates “*data chunks*” based on a “*data set*” and encrypts “*data chunks*” with respective encryption keys; the ’854 patent first distributes a “*data set*” into “*shares*” and then “*data chunks*” are generated by encrypting each “*share*” with a respective encryption key. In other words, the ’609 patent’s claimed “*data chunks*” are the ’854 patent’s claimed “*shares*.”

² I note that the method claims of Patent Owner’s ’609 patent (IPR2024-00214) have a similar “storing” limitation, but the limitation in the ’609 patent is broader because the ’609 patent’s “storing” limitation requires storing only “*at least one*

44. As seen above, limitation [1B] and its sub-limitations recite the steps of generating “data chunks” by “distributing” a data set into a plurality of “shares” and then encrypting each of those individual shares with its own “distinct encryption key.” In turn, limitations [1A] and [1C] recite the use of an external key to “perform[] an encryption operation” that “further secure[s]” the data chunks, and limitation [1D] recites storing the data chunks on different storage devices and storing with each data chunk “data indicative” of at least one of [1B]’s “distinct encryption keys.” As Ground 1 demonstrates (§VII below), in my opinion, each step recited in claim 1 was either disclosed in Orsini or POSA had reason to include in Orsini’s method. As Ground 2 demonstrates (§VIII below), in my opinion, a method comprising the steps recited in claim 1 would have also been obvious to a POSA over the combination of Foster and Hayhurst.

45. In my opinion, the dependent claims recite conventional data-encryption features. For example, dependent claims 4-6 recites methods for parsing the data set into shares, *e.g.*, using a random or pseudorandom technique. Dependent claims 8-10 recite methods for splitting an encryption key based on the

*data chunk ... on at least **one** storage device,”* whereas the ’854 patent’s “storing” limitation requires storing a “**plurality** of data chunks ... on a **plurality** of different storage **devices**.”

external key, distributing the data based on that split key, and encrypting and recovering data shares based on the external key. As I discuss in more detail below (§§VII.C-VII.K, §§VIII.E-VIII.M), these and other conventional features added by other dependent claims were well-known in the prior art, and POSAs had reason to implement them in encryption systems.

VI. CLAIM INTERPRETATION

46. I understand that determining whether a claimed invention is novel and/or non-obvious requires comparing the prior art to the claim. I understand that in some scenarios, an express construction defining the outer bounds of one or more claim terms may be appropriate during an IPR proceeding—for example, where the prior art would meet a challenged claim under one proposed construction but not another construction. In this IPR proceeding, I have been asked to provide my opinion concerning whether certain limitations are met under the parties’ preliminary claim constructions in the related district court litigation (EX1064, EX1065) and/or under the claim scope that Patent Owner appears to ascribe to the claims in its infringement contentions (EX1009). In my opinion, Petitioner’s proposed constructions (EX1064) are reasonable interpretations of the claim terms to a POSA at the time of the alleged invention. For my opinions regarding anticipation and obviousness, it is unnecessary to further analyze which party’s proposed construction most correctly sets forth the exact bounds of the

ordinary meaning of any claim term for which either party has proposed a construction, because I have applied each party's construction to every construed term (and I have applied the ordinary and customary meaning to any term for which neither party has proposed a construction in the district court) and, for the reasons discussed in the Grounds below, it is my opinion that the challenged claims are met by at least one combination of the prior art either way (*i.e.*, under either party's proposed constructions).

VII. GROUND 1: CLAIMS 1-10 ARE ANTICIPATED OR ALTERNATIVELY OBVIOUS OVER ORSINI

47. I have been asked to provide my opinion concerning whether claims 1-10 of the '854 patent are anticipated by Orsini or would have been obvious to a POSA in light of Orsini and the knowledge of a POSA. For the reasons explained below (§§VII.B-VII.K), it is my opinion that each of claims 1-10 is anticipated by Orsini and/or would have been obvious to a POSA over Orsini.

A. Orsini (EX1005)

48. I understand that Orsini is Patent Owner's own published patent application and shares an assignee and has overlapping inventors with the '854 patent. Orsini published on March 11, 2004 (code (43)), which I understand means it is prior art to the '854 patent even if the '854 patent were entitled to claim priority to its earliest-alleged priority date of November 18, 2005. Although Orsini was not cited during prosecution, I understand that Orsini issued as U.S. Patent No.

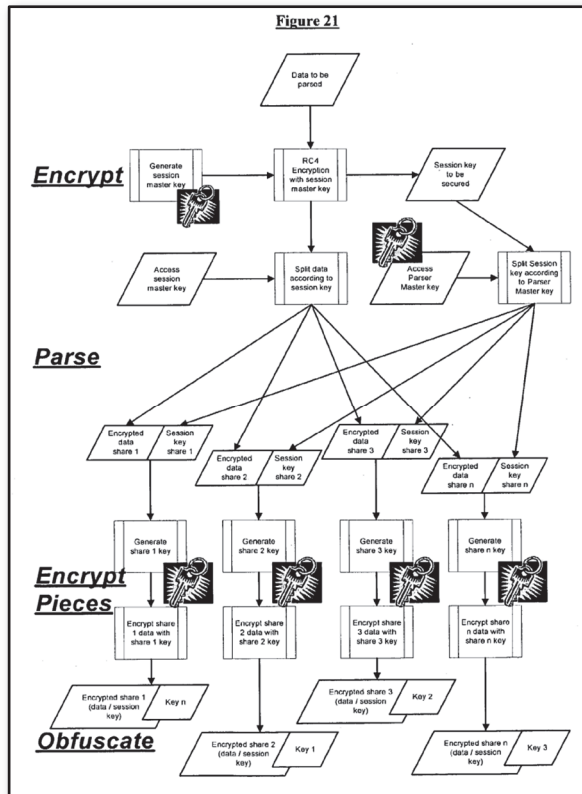
7,391,865, which was listed in an Information Disclosure Statement in the '854 patent's prosecution history. *See* EX1002, 66. However, Orsini and the '865 patent were not relied upon as the basis for rejection during prosecution of the '854 patent. *See generally* EX1002. I understand that the challenged '854 patent and Orsini are not formally related, but significant portions of their disclosures are identical (except for minor formatting, typographic and other minor differences). In particular, Orsini's disclosure in paragraphs [0001]-[0423] and Figures 1-25 are substantively identical with the '854 patent's disclosures in 1:1-66:3 and Figure 1-25. This is shown in EX1010, which is a PDF I requested to be generated that compares in "redlines" the text of the written description of Orsini to the '854 patent (*see* EX1010, 1-81) and shows side-by-side their respective Figures 1-25 (*see* EX1010, 82-107). The written descriptions used to make the comparison in EX1010 were obtained from the USPTO's Patent Public Search tool (<https://ppubs.uspto.gov/pubwebapp/>).

49. As EX1010 shows, the text of the written descriptions of Orsini and the '854 patent are identical except for some typos (*see, e.g.*, EX1010, 24), formatting changes (*see, e.g.*, EX1010, 26-27), changes to reference to related applications (EX1010, 2), and various other revisions that do not impact any of my opinions regarding the substance of their overlapping disclosures (*see* EX1010, 2,

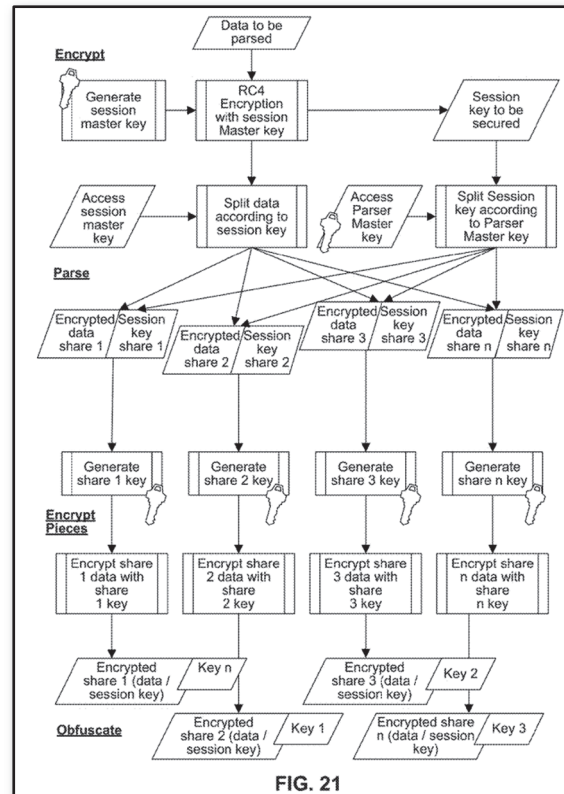
3, 7, 10, 52, 54-58, 60-65).³ EX1010 at 82-107 show Figures 1-25 of Orsini (left-side) and the '854 patent (right-side) side-by-side and demonstrate that these figures are substantively identical and only include formatting differences (*e.g.*, orientation of figures; changes to certain graphics, etc.).

50. Like the '854 patent, Orsini discloses “a method and system for securing sensitive data from unauthorized access or use.” Orsini (EX1005), Abstract. Orsini’s Figure 21 (below left) illustrates an exemplary encryption method (Orsini, [0293]) that is identical to the '854 patent’s Figure 21 (below right) except for formatting changes.

³ EX1010, 65-81 shows the primary text in the written description of the '854 patent that was not verbatim (or nearly verbatim) in the text of the written description of Orsini.

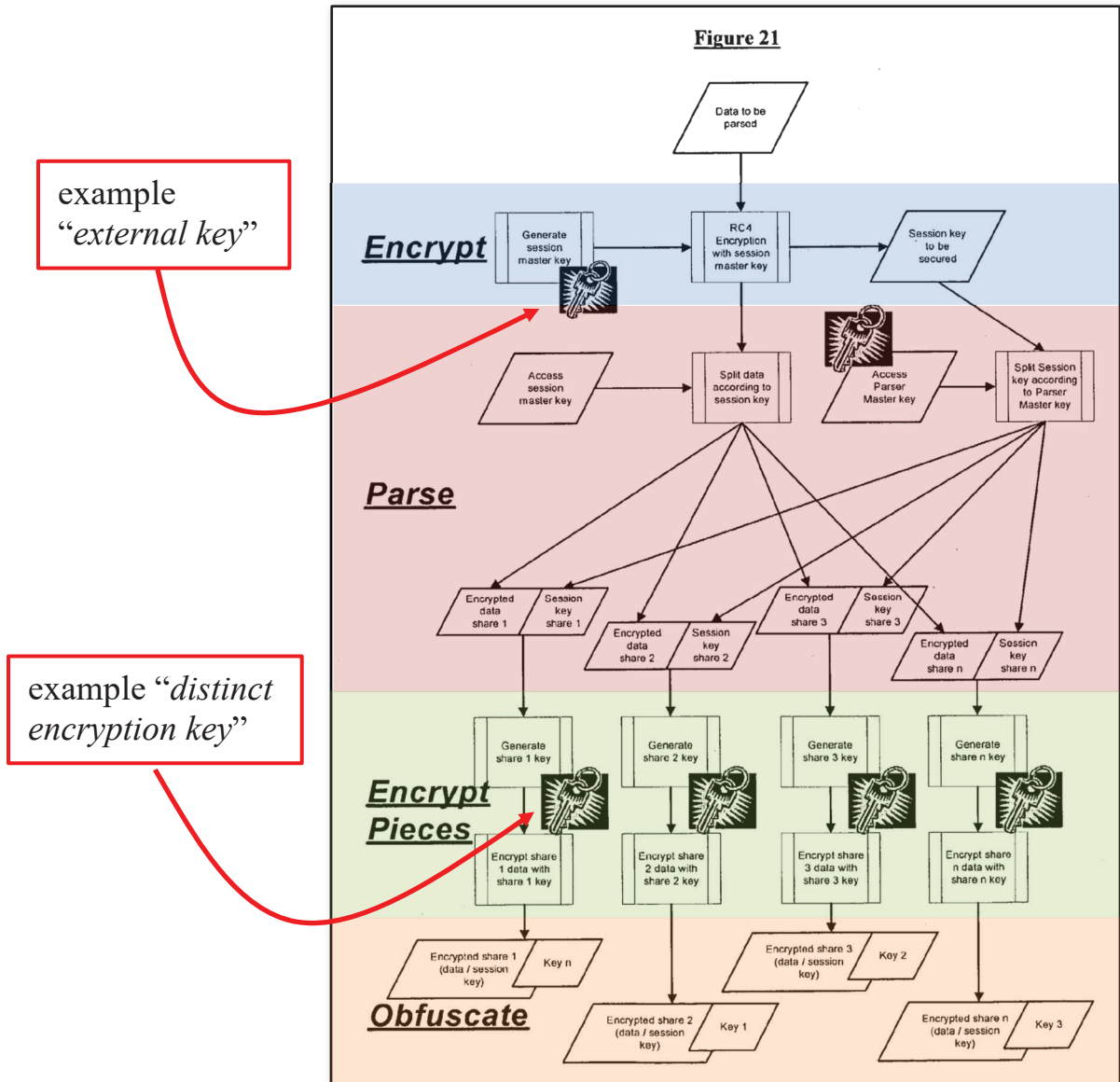


Orsini



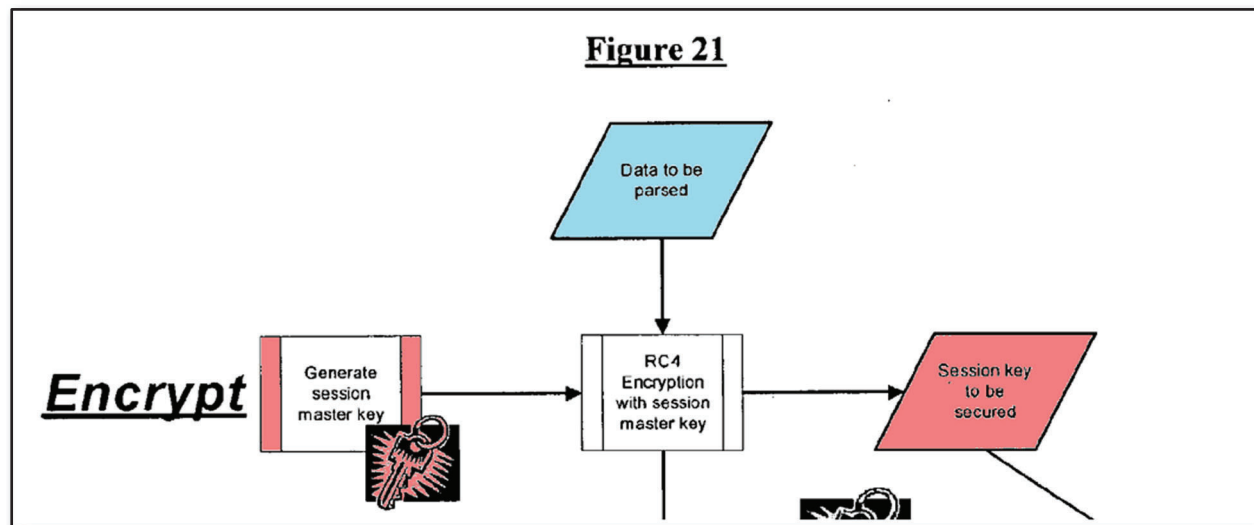
'854 Patent

51. As seen below (with color shading added), Figure 21's exemplary method has four basic steps – (1) encrypt, (2) parse, (3) encrypt pieces, and (4) obfuscate – which use various encryption keys, including a “session master key” (an exemplary key that meets limitation [1A]’s “*external key*,” for the reasons I discuss below in §VII.B.2.a.i) and multiple “share keys” (which meet limitation [1B2]’s “*distinct encryption keys*,” for the reasons I discuss below in §VII.B.5):



Orsini's FIG. 21
(with annotations and color shading added)

52. In Orsini’s “encrypt” stage (below with color added), a data set (blue) is encrypted with a “session master key”⁴ (red) (Orsini (EX1005), [0294]-[0295]):



Encryption with the session master key (and with most of the other keys Orsini discloses) can use any “well known” encryption algorithm can be used to encrypt data with the session master key.⁵ Orsini (EX1005), [0371], [0294]-[0295]. Orsini

⁴ POSAs understood “master keys” are encryption keys used to derive other encryption keys. For example, EX1051, which discloses at [0037] that it was known that “some systems rely on a master key to generate derived keys, which are then used to encrypt the data.”

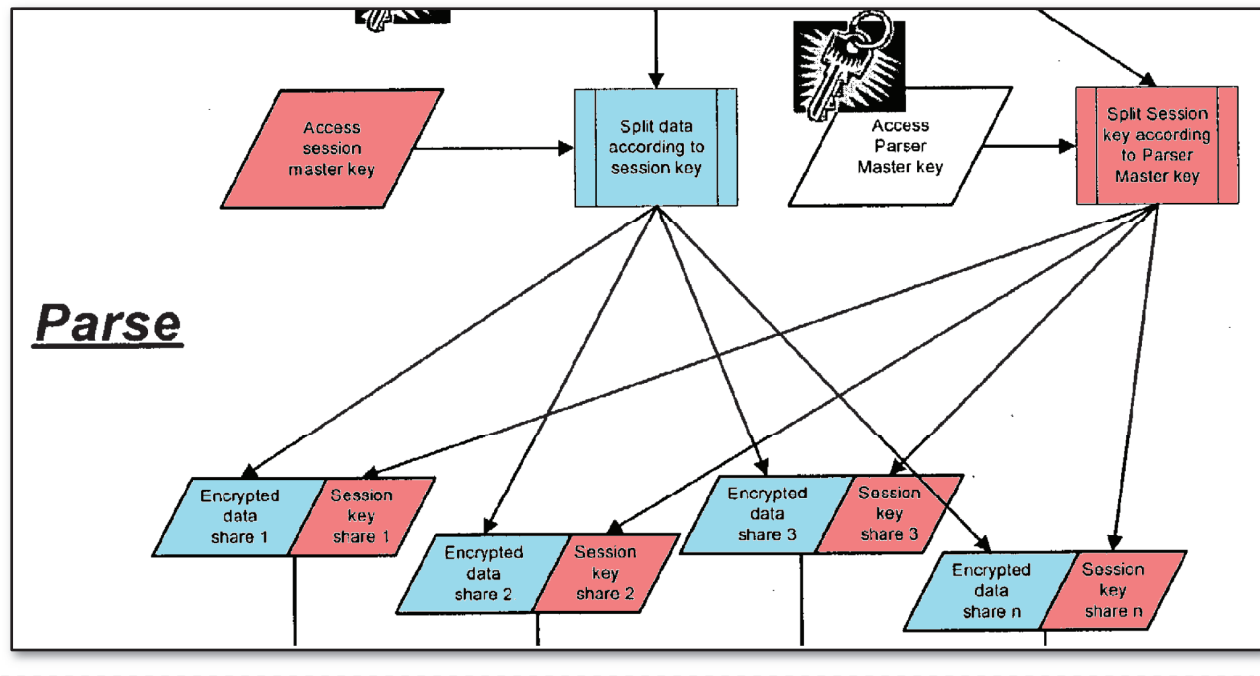
⁵ Figure 21’s first stage uses the RC4 encryption algorithm – from “RSA Security, Inc.” – however, Orsini states a “wide variety of encryption methodologies are

discloses “external session key managements” (Orsini, [0293]), which POSAs understood includes the creation, distributing, authenticating, and storing of session keys, like the session master key, externally. *See, e.g.*, EX1015, [0030], [0044] (Glossary section noting that “Key Management” refers to “[t]he various processes that deal with the creation, distribution, authentication, and storage of keys”).

53. Orsini’s “parse” stage splits the encrypted data set “into four shares or portions of parsed data according to the pattern of the session master key.” Orsini (EX1005), [0296]. In embodiments based on Figure 21, the session master key itself is split “according to the pattern of the Parser Master Key,” which is yet another encryption key. Orsini (EX1005), [0297], [0391]-[0392]. As seen below in an (color-added) portion of Figure 21, each portion of the split encrypted data (blue “data shares” below) is then paired with a portion of the split session key (red “key shares” below) into separate shares:⁶

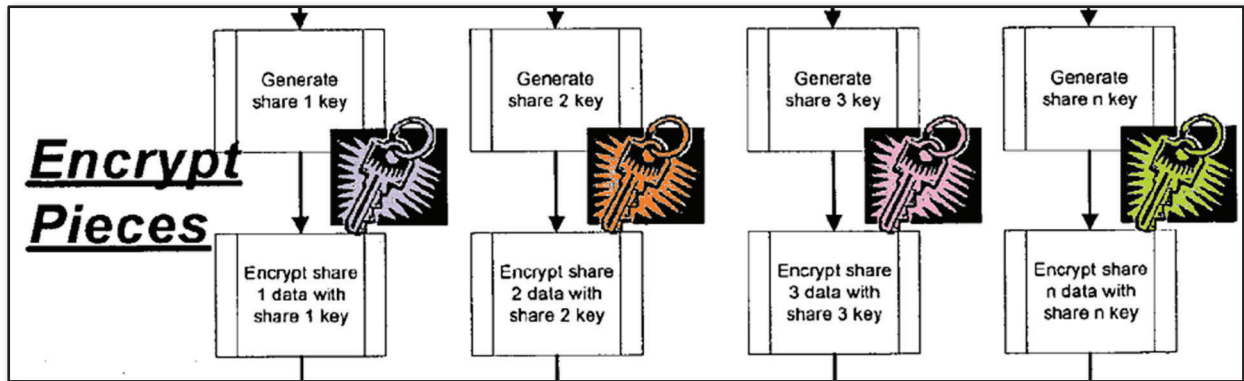
suitable.” Orsini (EX1005), [0335], [0371], [0378]; ’854 patent (EX1001), 59:37-39, 61:16-46 (same).

⁶ In other (similar) embodiments, the session master key is not split and paired with shares, but is instead stored separately from shares (which are stored with



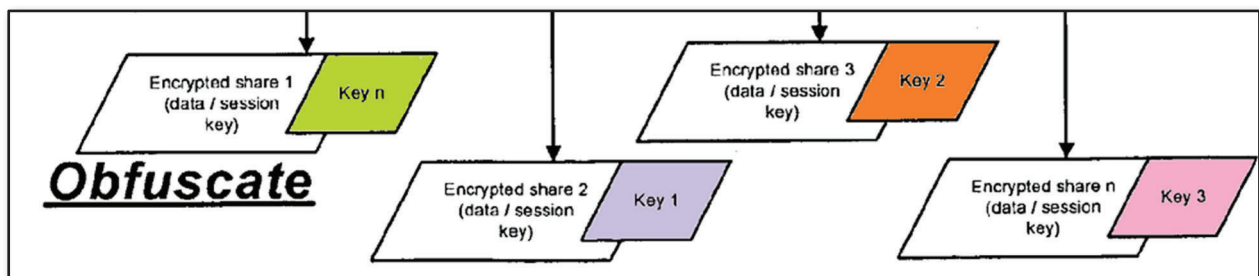
54. In the “encrypt pieces” stage, each share pair – each blue-red combination above that illustrates an encrypted share of the data that is paired with a share of the session key – is encrypted with its own distinct encryption “share key” (purple, orange, pink, and green below), resulting in an encrypted share that includes a portion of the (encrypted) original data set and a portion of the original session master key. Orsini (EX1005), [0298]-[0299], [0395]-[0396]. This is depicted below (with color annotations).

transaction IDs associated with the session master key). *See, e.g.,* Orsini, FIG. 24, [0323] (noting the “session master key data” is “stored” “in one or more *separate* key management table[s]”), [0323]-[0333].



As discussed above, in Figure 21’s embodiment, each encrypted share includes (1) a portion of the (encrypted) original data set and (2) a portion of the original session master key. However, as I discussed above, in other embodiments the session key is not split and paired with data shares. In either case, the encrypted data share – the encrypted portion of the original data set – meets [1B]’s “*data chunks*” for the reasons I discuss below in §VII.B.6.

55. Finally, in the “obfuscate” stage, each encrypted share is paired with one of the share keys used to encrypt a *different* share (e.g., encrypted share 1 is paired with share n key (green), encrypted share 2 is paired with share 1 key (purple), etc.). Orsini (EX1005), [0299]. This is depicted below:

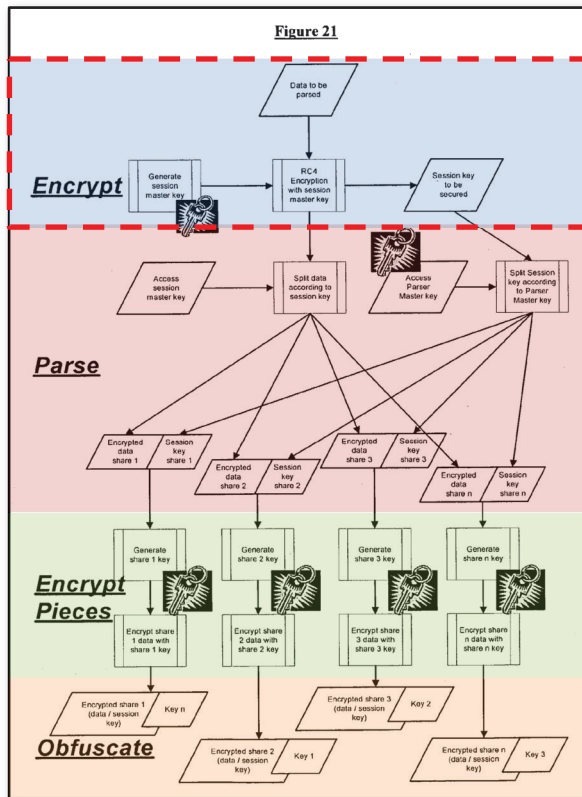


The different pairs of encrypted share (which may include a portion of the split session master key) and share key can then be stored, for example, in different data depositories, which may be in different physical locations. Orsini (EX1005), [0299], [0379].

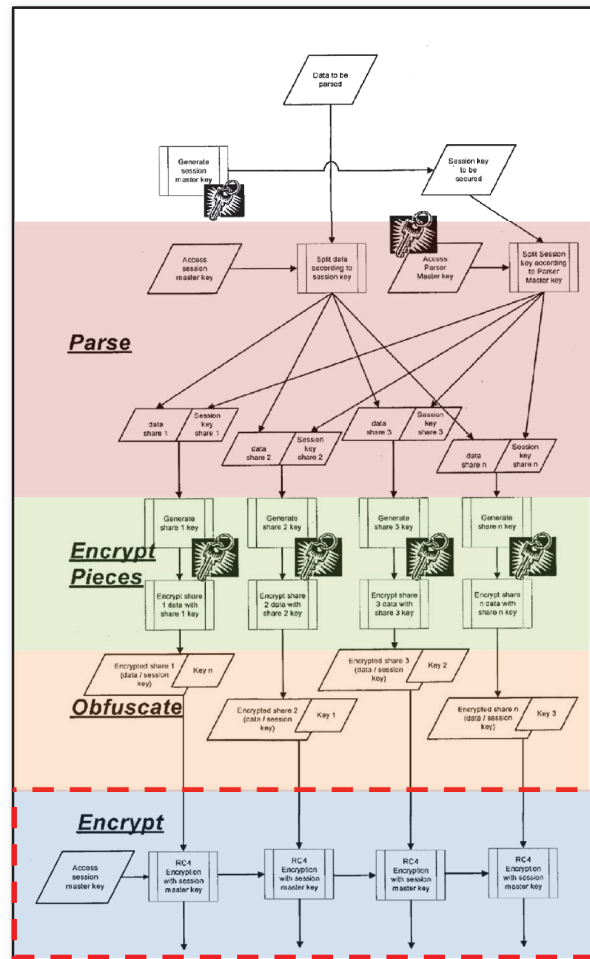
56. As I discuss more in §VII.B below, the '854 patent's claim 1 performs the same encryption steps illustrated in Orsini's Figure 21 but does not require (or exclude) encrypting the data set *before* splitting it into shares. Instead, claim 1 recites an encryption step with an encryption key (specifically, an “*external key*”) that occurs *after* a step corresponding to the “encrypt pieces” stage discussed above. Ground 1 demonstrates that Orsini meets the ordering of steps required by claim 1, including because Orsini discloses “that certain steps of the methods described herein may be performed *in different order, or repeated multiple times*, as desired.” Orsini, [0301].

57. One exemplary different-order Orsini embodiment that meets claim 1 in my opinion (for the reasons I discussed in §VII.B.7.a.i below) is conceptually illustrated below (right), where the “encrypt” step (top-left in Figure 21's embodiment) is reordered to be the last step before storing encrypted data (as illustrated in the figure on the right):

Original



Example Different-Order Version



58. Beyond the disclosure in Orsini's written description, Orsini also claims that reorder and/or repeat encryption steps. For example, Orsini's dependent claim 29 recites a method of encrypting data that includes the steps of (a) encrypting a data set, (b) splitting the encrypted dataset into two or more portions, (c) encrypting those split portions, and then (d) encrypting the encrypted split portions again using a different encryption algorithm from the encryption algorithm used at step (c). Orsini, claim 29; *see also* [0013] ("encrypting step may be repeated for one or more portions of the data"), [0423] ("encryption steps may,

in one embodiment, be repeated to produce a multi-layered encryption scheme”); claim 12 (reciting method that repeats encryption step “one or more times”).

Orsini thus teaches additional encryption step(s) that occur after individual shares have already been encrypted.

59. In my analysis below, I discuss implementations of Orsini’s encryption techniques that are disclosed or rendered obvious by Orsini. In most cases, the conventional encryption techniques I discuss below are complementary such that they all (or any desired subset thereof) can be performed in a single embodiment within the scope of Orsini’s disclosure. To the extent any of the conventional encryption techniques I discuss below are not compatible with being executed in the same encryption method, the selection of one technique over the other would have been a routine design choice for a POSA to select among the available encryption methods based on respective tradeoffs, as I discuss below, where relevant.

B. Claim 1:

1. 1[PRE] A method for securely storing a data set, the method comprising:

60. To the extent the preamble of claim 1 is limiting, Orsini discloses a “*method for securely storing a data set,*” as claimed.

61. For example, Orsini discloses “a method for *securing* virtually any type of data from unauthorized access or use,” and a step in the method is “*storing*

[] parsed, split or separated *data that has been encrypted.*” Orsini, [0013]; *see also id.* Abstract (noting that Orsini discloses methods “for *securing* sensitive *data* from unauthorized access or use”). Indeed, Orsini’s claims recite methods for “encrypting a *data set*” and then “storing” the encrypted data in data depositories. *See, e.g.,* claims 1, 10, 20, 30. POSAs understood that storing data in an encrypted state, as Orsini discloses, is a way of securely storing data.

62. In an exemplary embodiment, Orsini’s method is executed by a computer system (*e.g.,* a server) that executes, among other components, “a parser program” that performs “encryption” “on any *data set.*” Orsini, [0278]; *see also* [0013] (“This method may be incorporated into the operations of any computer, server, engine or the like, that is capable of executing the desired steps of the method.”). Although I understand that the parties dispute the definition of a “*data set*” (*see* EX1064, 6-7; EX0165, 9-10), because Orsini uses the identical term (*e.g.,* Orsini, [0278]) in a specification that shares significant portions of its disclosure (*see* EX1010) and inventors with the ’854 patent, POSAs understood Orsini’s use of “*data set*” meets the claim language regardless of which party’s definition is adopted.

2. [1A] receiving an external key from an external storage system,

63. As I discuss below (§VII.B.2.a), Orsini discloses receiving an “*external key*” from an “*external storage system,*” under both parties’ constructions

of “*external key*” and Google’s construction of “*external storage system*.”⁷ See EX1064, 11-13; EX1065, 8-9. Alternatively, in view of Orsini’s disclosure and a POSA’s general knowledge of encryption and storage techniques, POSAs had reason to implement Orsini’s encryption method to include a step of “*receiving an external key from an external storage system*,” under both parties’ construction, as I discuss in §VII.B.2.b below.

a. Orsini discloses [1A] (“*receiving an external key from an external storage system*”)

64. I have not been asked to opine on whether the ’854 patent provides reasonable certainty about the distinction between an *internal* key (or storage system) and an *external* key (or storage system). However, as I noted above (§VI) both parties have proposed explicit constructions of “*external key*” in related district-court litigation. As I discuss below in §VII.B.2.a.i, in my opinion, Orsini encryption methods using “external session key management” meets [1A] under SFI’s proposed construction of “*external key*.” As I discuss below in §VII.B.2.a.ii, in my opinion Orsini’s encryption methods using public encryption keys meets [1A] under both parties’ proposed constructions of “*external key*” and Google’s proposed construction of “*external storage system*.”

⁷ Based on EX1065, it is my understanding that Patent Owner has not proposed a separate construction of “*external storage system*.”

i. Encryption keys received from and stored at Orsini's "external" key management

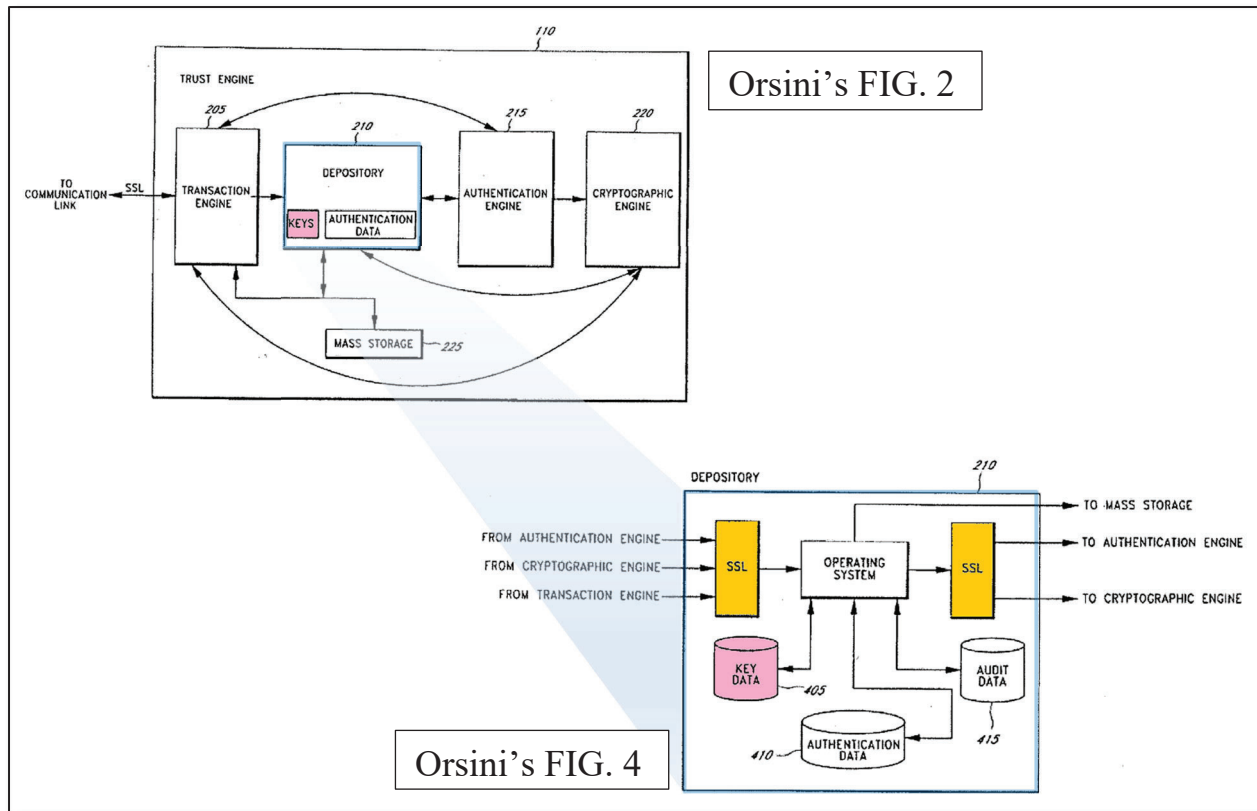
65. I understand that Patent Owner argues that "[n]o construction [is] necessary" for [1A]'s "*external key*" or in the alternative construes the term to be "a key stored separately from the data chunks." EX1065, 8-9. As discussed below, in my opinion Orsini meets [1A] under either of Patent Owner's alternative constructions.

66. While Orsini's Figure 21 shows its "parser software suite" generating the session master key (Orsini, [0293]-[0295]), Orsini does not specify at what physical device in Orsini's system the generating occurs (whether its cryptographic engine 220 or another component, like the depository 210, which I discuss below). Even if some embodiments are generating the session master key at the encryption device (*e.g.*, the cryptographic engine) rather than receiving it from a storage system, Orsini discloses embodiments where its encryption techniques "may be implemented with *external session key management*." Orsini, [0293], *see also* [0329] ("passing the Session Master Key ... to the calling program for *external management*"); [0381]-[0386] (disclosing desirability of keeping some the session master key (and other "elements") "physically separated from the remaining components of the system (under the control of a different system administrator for example)").

67. POSAs understood “Key Management” generally refers to “[t]he various processes that deal with the *creation, distribution*, authentication, and *storage* of keys.” EX1015, [0030], [0044]. Thus, POSAs understood that instead of the session master key being internally generated, Orsini’s “external session key management” can be created, distributed, and stored by an external session key management for later use by the system implementing Orsini’s encryption methods.

68. Orsini illustrates an exemplary storage system for encryption keys as “depository 210” in Orsini’s Figures 2 and 4 (which I have annotated below). In Figure 2 (annotated below left), cryptographic engine 220 and depository 210 are part of “trust engine 110,” which “comprises one or more secure servers.” Orsini, [0034], [0036], [0066]. Orsini’s parser program (which performs the encryption method I discuss below in §§VII.B.3-VII.B.8) is implemented in cryptographic engine 220. Orsini, [0090] (“cryptographic engine 220 [] include[s] a data splitting module”), [0278] (“the data splitting module comprises a parser program...which comprises data splitting, encryption, and decryption”).

69. Depository 210 – illustrated in more detail in Figure 4 (annotated below right) – “stores cryptographic keys” – specifically, “data 405” (shaded pink below) in the depository “correspond[s]” to the cryptographic keys. Orsini, [0085].



Orsini's FIGS. 2, 4 (annotated)

POSAs would have understood from these disclosures that encryption keys used by cryptographic engine 220 (e.g., the session master key discussed in §VII.A above) are stored in and received from depository 210 (e.g., as key data 405).

70. POSAs would have further understood from Orsini's disclosure that cryptographic engine 220 and depository 210 are stand-alone components implemented on different "secure servers" (Orsini, [0066]) that are in communication with each other and the depository is not built into the cryptographic engine. Orsini, [0066] ("trust engine 110 *comprises one or more secure servers* for accessing and storing sensitive information"), [0079] (noting

that Figure 2 illustrates various engines that communicate with each other and depository 210 (that is, they're distinct components, rather than built into each other)).

71. In particular, Figure 4 depicts the cryptographic engine and depository communicating with each other using SSL (Secure Sockets Layer), a network-based security protocol (which is annotated in orange above). Orsini, [0079], [0095]. The fact that SSL is a network-based security protocol is corroborated by, for example, EX1011 at 5:19-45, which notes that “authenticated communication [on the ‘web’] can be realized through, for example, SSL” and at 16:43-45, which notes that “protocols for secure transmission of information over the Internet” include “SSL.” In addition to SSL, Orsini discloses that the communication between the cryptographic engine and depository “us[es] conventional HTTP routing techniques” and “URLs” (Uniform Resource Locators). Orsini (EX1005), [0081]. POSAs understood these protocols—HTTP/URL routing with SSL security—are conventionally used for communication between systems separated by different networks and/or the Internet and that operate independently, including by being operated by different entities.

72. My opinion in this regard is corroborated by EX1012 at [0052], which states that HTTP is a protocol for “web servers and browsers,” and at [0065], which states that SSL “is a protocol for transmitted private documents via the

Internet.” Similarly, EX1013 at 3:16-29 discloses “a controlled cryptographic key repository, such as a secure network or third party holder of cryptographic keys,” which corroborates my opinion that it was known to manage security keys in servers separated by secure networks and/or under the control of different entities.

73. POSAs understood from Orsini’s disclosure that to perform a cryptographic operation when keys are externally managed, the cryptographic engine receives encryption key(s) (*e.g.*, a session master key) from depository 210. For example, Orsini discloses that “cryptographic keys are generated at the cryptographic engine 220 and forwarded to the depository 210.” Orsini (EX1005), [0090]. Orsini further discloses that the depository 210 in Figure 4 “stores data 405 corresponding to the cryptographic keys.” Orsini (EX1005), [0095]. The “cryptographic engine 220 also comprises a cryptographic handling module 625” (Orsini, [0107]), which “*request[s] the matching private key be assembled from...depository 210*” and “use[s] the matching private key to perform cryptographic actions or functions” (Orsini, [0134]). Because cryptographic keys in general are stored at and received from depository 210, POSAs would have understood that session master keys are stored at and received from depository 210. Orsini, [0095] (depository 210’s “store[d] data 405 correspond[s] to the cryptographic keys”). *See also* §VII.B.2.b below (where I explain why receiving encryption keys from external storage systems was also obvious over Orsini).

74. Under Patent Owner’s view that [1A]’s “*external key*” requires no construction and Patent Owner’s infringement contentions, which purport to identify an encryption key received from a storage system that is (allegedly) separate from the system performing encryption (EX1009, 2), POSAs understood that Orsini’s session master key that cryptographic engine 220 receives from depository 210 (which is a separate “secure server” as I discussed above) is an “*external key*” under Patent Owner’s view.

75. Under Patent Owner’s alternative construction of “*external key*” as “a key stored separately from the data chunks” (EX1065, 8), POSAs understood Orsini’s session master key is stored separately from “*data chunks*” (encrypted shares, as discussed §VII.B.3 below) because (in some Orsini embodiments) the session master key can be requested from the depository 210 and used by the cryptographic engine (Orsini, [0134]) ***before*** the encrypted shares are generated, as I discussed §VII.A above in describing some of Orsini’s embodiments.

76. If Patent Owner’s alternative construction of “*external key*” also requires the “*external key*” to be stored separately from data chunks ***after*** data chunks have been generated and stored, Orsini discloses this as well. For example, Orsini discloses embodiments that “store the session master key data in one or more separate key management table[s]” (Orsini, [0302]), which POSAs understood meant the session master key is stored separately from data encrypted

with the session master key (as illustrated in Orsini’s FIG. 22, where a transaction ID, rather than a portion of the session key, is paired with each encrypted data share). *See also* Orsini, [0303]-[0308], [0323]-[0333], FIG. 24.

77. Orsini also discloses that embodiments can “stor[e] the encryption master key data *separately from the data*” encrypted with that master key (Orsini, [0055], [0057]), for example, as illustrated in Orsini’s FIG. 22. *See also* Orsini, [0302] (“store the session master key data in one or more *separate* key management table[s]”), [0303]-[0308], [0323]-[0333], FIG. 24. These embodiments meet the challenged claims as the rest of Ground 1 below demonstrates (§§VII.B.3-VII.B.8), because the session master key’s storage location does not affect the method steps’ mapping to the other claim limitations that I discuss in the sections below.

ii. Public-key encryption keys received from a certificate authority

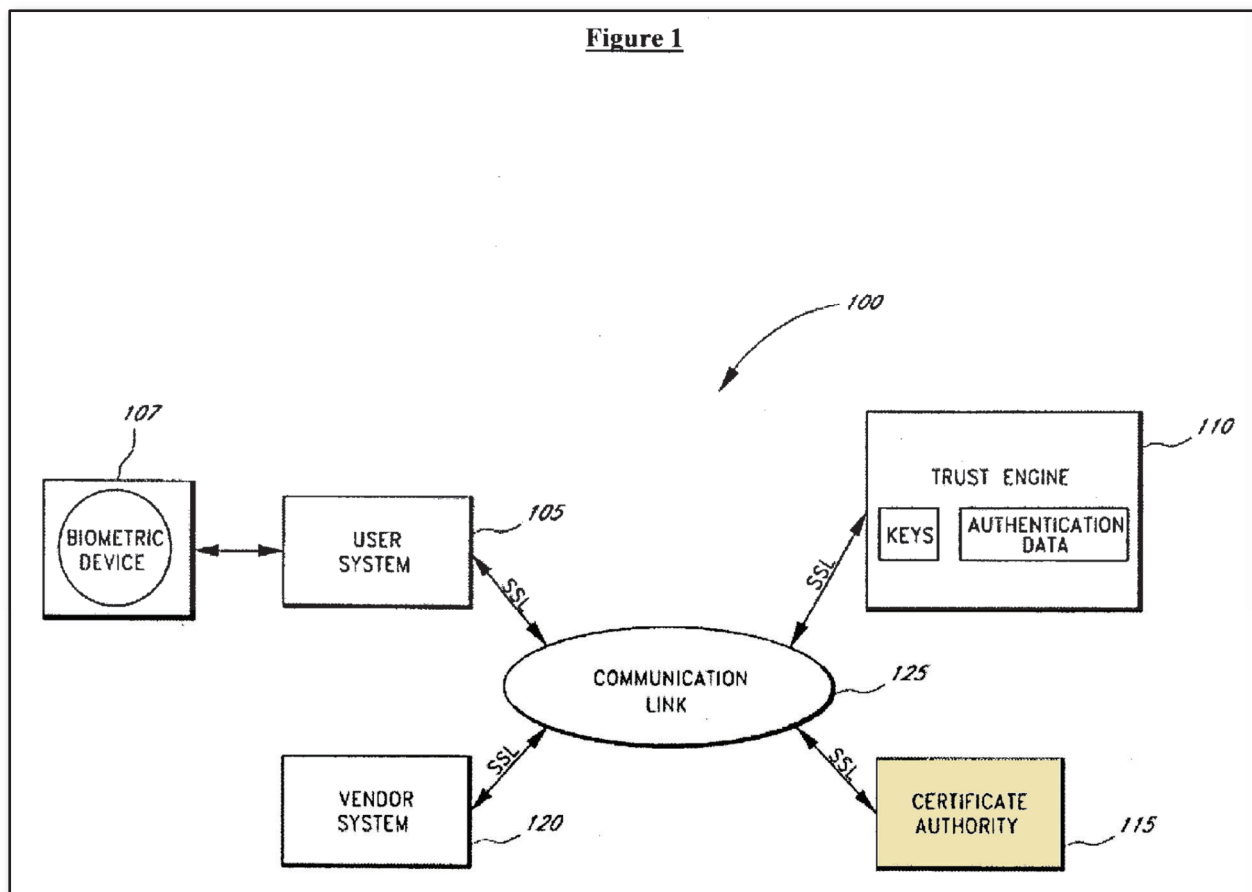
78. Alternatively, Orsini discloses using public encryption key that are issued by a certificate authority (Orsini, [0125], [0007]), which meets [1A] under both parties’ proposed constructions, for the reasons I discuss below.

79. Orsini discloses that the “certificate authority 115 generally issues digital certificates,” which “include[] a *public key* of the user or system.” Orsini, [0125], [0007]. As Orsini states, a “public key system” is a “popular cryptographic system” “that uses two keys, a public key known to everyone and a private key

known only to the individual or business owner” and that “data encrypted with one key is decrypted with the other and neither key is recreatable from the other.”

Orsini (EX1005), [0007]. Thus, a public key in a certificate issued by a certificate authority is an encryption key.

80. Orsini’s Figure 1 (which I have annotated below) illustrates a certificate authority 115 communicating with trust engine 110 (which is the overall system executing Orsini’s encryption methods) over, *e.g.*, the Internet via SSL links (which are links I discuss above in §VII.B.2.a.i). Orsini (EX1005), [0062], [0076] (disclosing that FIG. 1’s “communication link 125 preferably comprises the Internet”), [0077]-[0078] (noting that the communication link can be over the “web” or various other networks, *e.g.*, “telephone networks”), [0083] (disclosing use of “conventional SSL encryption technologies”).



Orsini's FIG. 1 (annotated)

Orsini expressly discloses that the certificate authority is “*external* to the trust engine.” Orsini, [0130] (noting the “proprietary certificate authorities internal or *external* to the trust engine”). As Orsini’s Figure 2 illustrates, trust engine 110 includes cryptographic engine 220, which executes Orsini’s encryption method (as I discussed above in §VII.B.2.a.i). A public key received from the certificate authority therefore meets the interpretation of “*external key*” in Patent Owner’s infringement contentions (which I discussed above in §VII.B.2.a.i).

81. For the same reason the certificate authority is “*external*” to the trust engine, the certificate authority is “*external*” to the storage system(s) within the trust engine that store “*data chunks*” (which I discuss below in §VII.B.8). For example, as shown in Figure 1 above, the trust engine processes “authentication data,” which is stored in depository 210. Orsini, [0095], FIG. 2 (illustrating that trust engine 110 includes cryptographic engine 210); *see also* [0109]-[0110] (disclosing a “depository system” with “data storage facilities” where “sensitive data” is “stored”), [0111] (“each data storage facility...compromises a...storage system”). Thus, a public key stored at and received from the certificate authority meets Patent Owner’s alternative construction of “*external key*” because it is “a key stored separately from the data chunks.” EX1065, 8.

82. Orsini discloses that the certificate authority is controlled by “a trusted *third-party* organization or company.” Orsini, [0072]. POSAs were also aware that certificate authorities “can also issue public-private cryptographic key pairs.” EX1067, [0004], [0016]. As part of issuing these keys, the certificate authority would generate the keys (*e.g.*, for entities that needed many keys). Thus, a public key generated by and received from the third-party certificate authority meets Google’s construction of “*external key*” because it is “a key generated by a third party.” EX1064, 11-12.

83. POSAs understood from Orsini’s disclosure that the certificate authority would have stored the public keys it issued and/or managed in a storage system (in order to distribute the keys upon request, which may occur at unpredictable, various times). My opinion in this regard is corroborated by Orsini at [0128], which notes that third-parties are able to “request[] the user’s digital certificate” (which comprises a public key as I discussed above) from a certificate authority. Thus, third parties can make the request for the digital certificate at different times, and so there would have been reason to store the certificates in a storage system rather than have to regenerate them upon every request.

Alternatively, even if the certificate authority were not using a storage system, a storage system to store its public keys would have been obvious as the conventional and beneficial way for the certificate authority to manage public encryption keys. My opinion is corroborated by the Orsini discloses discussed above in this paragraph and by EX1014 at 2:36-63, which describes as background information regarding the “public key infrastructure” that a certificate authority “*stores* the public key in a database” (*e.g.*, a database of a storage system) and that “[t]he public key database is maintained by the certificate authority.”

84. Because the certificate authority can be a third party, its storage system meets Google’s proposed construction of “*external storage system*,” which is “a storage system controlled by a third party.”

85. Accordingly, public encryption keys issued by a certificate authority meets both parties’ construction of “*external key*” (EX1064, 11-12; EX1065, 8-9) and, the certificate authority controlled by a third party entity and its storage systems meet Google’s proposed construction of “*external storage system*” (EX1064, 12-13).

86. Beyond “*receiving*” the “*external key*,” claim 1 also recites “*performing an encryption operation based on the external key to further secure the plurality of data chunks*” (limitation [1C]). As I discuss below (§VII.B.7.a.iii, [1C]), Orsini discloses “us[ing] the public key of the user, for example, to encrypt the session key that will be used to encrypt” data. Orsini, [0172]. As I also discuss below (§VII.B.7.a.iii, [1C]), POSAs separately had reason to further secure data chunks with a public key received from the certificate authority.

b. Alternatively, [1A] would have been obvious over Orsini

87. Claim 1 recites that [1A]’s “*external key*” is involved in step [1C] to “*perform[] an encryption operation...to further secure the plurality of data chunks*” (which I discuss below in §VII.B.7). As I discussed above (§VII.B.2.a.i), Orsini’s cryptographic engine receives a session master key, which is an “*external key*” under Patent Owner’s proposed constructions. In Sections VII.B.7.a.i-VII.B.7.a.ii below, I demonstrate how Orsini’s session master key is used to meet limitation [1C]. Alternatively, in Sections VII.B.7.b.i-VII.B.7.b.iv below, I discuss

how other conventional encryption keys that are disclosed and/or rendered obvious by Orsini's disclosure also meet limitation [1C].

88. If Patent Owner argues that Orsini does not explicitly disclose receiving the session master key or any of the other conventional encryption keys discussed in Sections VII.B.7.b.i-VII.B.7.b.iv below as an “*external key*” from an “*external storage system*,” in my opinion POSAs had multiple reasons (which I discuss below) to implement Orsini to receive any of those keys as an “*external key*” from an “*external storage system*” under both Google's proposed construction of “*external storage system*” and under Patent Owner's infringement contentions, which identifies key allegedly “stored centrally in Keystore, a repository [allegedly] built specifically for storing keys.” EX1009, 2.

89. First, as I discussed above (§VII.B.2.a.i), Orsini provides “*external session key management*” and discloses a system with “multiple cryptographic engines” having a “single point of ... key management.” Orsini, [0180], [0176]-[0179], [0293], [0412]. POSAs would have recognized several benefits to implementing Orsini's single point of external key management for multiple cryptographic engines by centrally generating Orsini's session master keys (and any of the conventional encryption keys I discuss in §VII.B.7.b below) and storing them separately from data chunks in a centralized storage system built specifically

for storing encryption keys, so that those keys could be centrally generated, stored, and then distributed to (received by) Orsini's multiple cryptographic engines.

90. One benefit would have been eliminating the need (and associated complexity and cost) to install and maintain internal-key-generation and -storage mechanisms on each cryptographic engine. This would have also beneficially reduced the processing load on and storage demand on each cryptographic engine (by not requiring it to generate keys itself). POSAs knew that encryption key generation systems preferably comprise a secure source of entropy (randomness) to generate secure encryption keys, which could require substantial processing resources. POSAs would have understood that it was conventional for a key server to comprise those entropy sources and processing dedicated to generating secure encryption keys. My opinions in this paragraph are corroborated by (at least) the following:

- EX1020 (filed 2005), 1:34-38: disclosing as background knowledge that “**key generation**” is a “**processing intensive operation[]**”; and
- EX1021 (filed 2003), [0032]: disclosing that having encryption keys “stored on an **external key server**,” which makes “thereby making it **safe to store encryption/decryption keys on external storage** instead of in limited internal memory.”

- EX1055 (filed 2004), [0006]: disclosing as background information that “[s]ince the strength of an encryption key depends in large part on the unpredictability of the characters in the key, **generation of strong encryption keys requires the key generation program or network element to have access to a source of random numbers, referred to herein as an entropy source.**” EX1055 further notes (at [0009]) that “[a]s VPNs increase in size and number, generation of keys by the group members has been replaced by **centralized key management systems, referred to herein as Group Control Key Servers (GCKSs).** A GCKS may provide many services on the network as well as key generation and management.”
- EX1067 (filed 2003), [0016]: “[C]ryptographic keys can be securely stored and/or distributed outside of the HSM [hardware security module] in which the cryptographic key was generated. This *can reduce the amount of storage space needed for cryptographic keys within individual HSMs.*”
See also [0005] (“hardware security modules (HSMs)”).

91. A centralized key management system would have also facilitated Orsini’s stated goal of “integrat[ing] into existing infrastructure” (Orsini, [0280]), which may not have been capable of generating secure encryption keys because, as

I explained above, generating encryption keys is a “processing intensive operation[.]” EX1020, 1:34-38:

92. A centralized key management system would have also facilitated distributing, storing, and updating encryption keys shared by multiple users and/or multiple cryptographic engines (*e.g.*, group-level encryption keys, that I discuss below in §VII.B.7.b.ii).

93. My opinion that these (key) storage systems for encryption keys was a well-known and beneficial implementation of encryption key management systems is corroborated, for example, by the following:

- EX1016 (filed 2002), [0013]: disclosing “**secure key management whe[rein] separate devices are used for generating and utilizing the keys.** For example, a system that uses a cryptographic accelerator typically includes a **separate processor that generates the keys** (*e.g.*, a host processor). Thus, at some point in time the keys are sent from the host processor to the cryptographic accelerator so that the cryptographic accelerator can decrypt the received encrypted data.”;
- EX1017 (filed 1993), 1:65-2:2: “In more advanced secure communication systems, **encryption key variables are stored and distributed electronically to communication units using a key management controller or KMC.** This method **provides for rapid distribution of**

encryption key variables to the communications units, **reducing significantly the time it takes to load new encryption key variables** into all communication units in a system.”;

- EX1018 (filed 2004), 5:19-44: discussing a “data key data store 45 which contains data which can be used to find data keys assigned to users and user groups”; “it contains all the data keys which can be used to decrypt data in the application data store 37.” Further noting that the key data store can “**be set up centrally at a remote location**” to allow “it to be isolated from the data processing facility” and “**allows particularly stringent security precautions**” “such as particularly restrictive firewalls.”;
- EX1019 (filed 2001), [0034]: disclosing a “**key distribution center 140** and a group database server 150.” The key distribution server 140 “may be configured to provide **a secure method of transferring encryption/decryption keys within the system 100**” and “[t]he group database server 150 may be configured to provide central access to the user of the system 100 for information related to file encryption groups.”;
- EX1022 (filed 2002), [0004]-[0006]: disclosing as background information “us[ing] **central key management**, wherein the **keys are stored on a centralized server that is itself secured from access by attackers.**” And

that “[t]he use of **centralized key servers to secure data is known**. In a typical arrangement, the system that maintains data secured by the key (the ‘key client’) authenticates itself to the key server and the key server authenticates itself to the key client.” And disclosing a system where “secured data is stored on a first system secured by a control key stored securely on a key server.” And at [0015]-[0017], which discloses “**remote key server**” where “‘remote’ refers to being **separated by a security boundary** such that access to a local system does not automatically confer access to the remote system” and that “**physical separation** is one way to provide for remoteness.”; and

- EX1023 (filed 1996), 1:34-36: describing as prior art that the “[d]istribution of session keys is a complex matter. **Session keys** are usually transmitted encrypted by a master key and **often come from central key distribution center (KDC)**.”

94. A centralized key-storage system specifically built to generate, store, and distribute encryption keys would have beneficially reduced the risk of theft and/or sabotage by rogue insiders (*e.g.*, disgruntled employees) by preventing them from having direct access to the key-management system (*e.g.*, the key-generation algorithms), which would be maintained in a stand-alone component not built into

the parser program's component and in a secure location (separate from the "*data chunks*" stored in systems designed to store non-key data).

95. My opinion in this regard is corroborated by, at least, the following:

- EX1024 (filed 2001), [0003]-[0010]: describing as background information that issues with encryption key management that should be considered include that "a genuine user himself might not be able to access his information due to the loss or the damage of the decryption key," "a rogue employee may encrypt the critical information of a company and then request money for releasing the decryption key," and that "it sometimes happens that the government needs to have the right of an access to a key or plaintext with legal reasons such as the criminal investigation." Noting that ways to address these issues include "***a commercial KES that provides services such as key-backup is used,***" "a commercial KES that provides services such as key-backup is used," and that "KES is a typical solution" to having "a hidden trapdoor ... to prevent the cryptographic side effects."; and
- EX1025 (filed 1998) 4:21-59, FIGS. 1-2: "The common Achilles heel of prior cryptographic systems is the key management system, which cryptoanalysts have found to be an easier target for discovering encryption keys than attempting to crack the encryption itself" and that "[c]ontemporaneous encryption key management systems" included

“automated,” where “[a] user at computer 10 may through a user authentication process gain access to the server 12 and server storage 14, *but not to sensitive files stored in the key server system 17 which is accessed by the server 12 by way of a communications link 16.*”

96. Second, POSAs would have also understood that receiving Orsini’s encryption keys from a centralized key-storage system built for generating, distributing, and storing keys (separately from “*data chunks*”) would have been a simple substitution of one known element (from wherever encryption keys are allegedly stored and received from in Orsini if not from an “*external storage system*”) for another (a conventional, centralized key-management system).

97. Including evidence cited above, my opinion in this regard is corroborated by, at least, the following:

- EX1022 (filed 2002), [0003]-[0005]: disclosing as background information an example where “keys are stored on a centralized server that is itself secured from access by attackers,” which thereby keeps cryptographic keys secure and thus improving “security of the encrypted data.”;
- EX1027 (filed 2000), 3:27-35: disclosing that a “known [system] for key management includes a key-server,” which “is a single system that provides keys to individuals upon identification or authorization” and “is useful in

large organizations since it permits changing of system access codes without requiring every user to provide their personal key data storage device.”; and

- EX1027 (filed 2000), 6:18-41: disclosing “a simplified block diagram of a prior art method of storing key data on a main key-server 1 and of storing backup key data on a duplicate key-server 4,” where “main key-server 1 is provided with all security data relating to a network 2.”

98. For additional security (such as to better prevent the rogue-employee and sabotage problems that I discussed above), POSAs also had reason to implement the above-discussed key-management systems under the control of a third party (*e.g.*, in a secure, external facility under third-party control), which meets Google’s construction of “*external storage system*” as “a storage system controlled by a third party.” EX1064, 11-13. My opinion in this regard is corroborated by, for example, EX1024, [0003]-[0010] and EX1025, 4:21-59, FIGS. 1-2, which I discussed above in this section.

99. In my opinion, POSAs would have had a reasonable expectation of success in each of the above implementations of Orsini’s encryption key-management system. As I noted above, Orsini discloses that its “[e]ncryption key management... may be integrated into an existing infrastructure *or any other desired location.*” Orsini (EX1005), [0280]. Orsini further discloses that keys may be stored in “a *standard* key store” or “custom key store.” Orsini (EX1005),

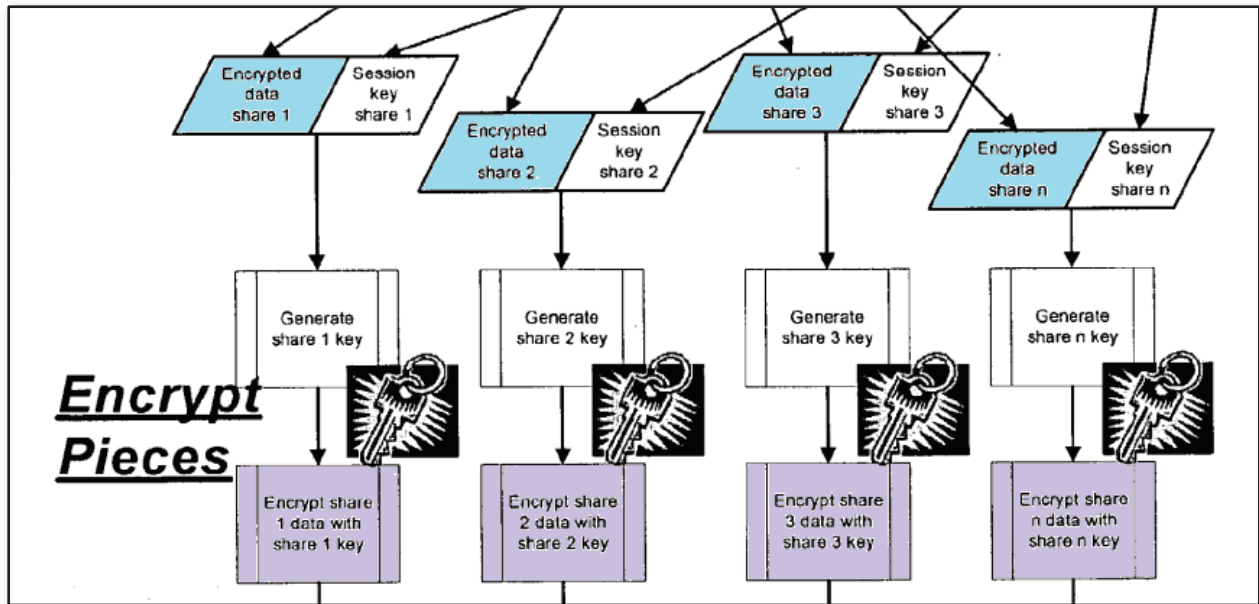
[0392]-[0394]. As I demonstrated above, known, standard key stores included centralized key-management systems specifically designed for generating, storing and distributing encryption keys. Thus, in my opinion, it would have required only a POSA's ordinary cryptographic-system design skill to have Orsini's cryptographic engine(s) receive encryption keys from any of the conventional key-management systems discussed above.

100. My opinion in this regard is independently and additionally corroborated by the '854 patent itself, which provides no details about, ascribes no benefits to, and identifies no challenges in implementing an external storage system for an external key. *See, e.g.*, '854 patent (EX1001), 77:58-63, 78:66-79:23, 79:66-80:4. Given the patent's silence on external storage systems, it would have reinforced a POSA's understanding that the '854 patent is relying on conventional, existing "external key stores" – rather than key stores invented or customized in any non-conventional, non-trivial way – to implement the '854 patent's conventional encryption and storage methods. This separately corroborates my opinions (1) that receiving external keys from external storage systems, as claimed, were by the earliest alleged priority date of the '854 patent conventional and required only ordinary skill to implement in Orsini and (2) that POSAs were fully capable of, and would have reasonably expected success in,

selecting between the known methods of storing and distributing encryption keys, which included from external storage systems, as I discussed above in this section.

3. [1B] generating a plurality of data chunks based on the data set, such that the data set can be reconstructed using at least a minimum number of the plurality of data chunks:

101. Orsini discloses splitting “the data into N **shares**,” where N is a plurality of (two or more) shares Orsini (EX1005), [0280]-[0281]. Orsini’s Figure 21, for example, shows at least 4 shares. In Orsini, and illustrated in Figure 21 (a portion of which I colored below), each **share** is a portion of the data set (and thus “*based on the data set*”) and is encrypted with a distinct encryption key (*e.g.*, “share 1 key,” share 2 key,” etc.). Orsini (EX1005), [0281], Claim 1 (claiming “b) separating the encrypted data set into two or more portions of data” and “c) encrypting one or more of the portions of data from step b)”). As I discuss below (§VII.B.6, [1B3]), the encryption of the plurality of shares with the distinct encryption keys generates a plurality of **data chunks (encrypted shares)** based on the data set, as shown below from Orsini’s Figure 21:



Orsini, FIG. 21 (partial; annotated)

POSAs thus understood Orsini to be “*generating a plurality of data chunks based on the data set*” under SFI’s construction – because Orsini is “creating a plurality of data chunks based on the data set” for the reasons I discussed above. EX1065, 2-3.⁸

102. Orsini also discloses providing “redundancy in the **shares** such that only a **subset** of the **shares** are needed to reassemble or restore the data to its original or useable form,” that is, to reconstruct the data set. Orsini (EX1005),

⁸ I understand that Google has not proposed a construction of “*generating a plurality of data chunks based on the data set*” in the related district-court litigation. See EX1064.

[0282]-[0284].⁹ For example, Orsini discloses that a data set can be split into four **shares** such that “only three of the four **shares** are necessary to reassemble or restore the data.” Orsini (EX1005), [0283]. The same number of **data chunks** (encrypted shares) would also be required to reconstruct the data set because each **share** is encrypted into a respective **data chunk**, as I noted above and discuss further below (§VII.B.6, [1B3]). Thus, Orsini discloses that “*a data set can be reconstructed using at least a minimum number of data chunks*” under both parties’ proposed constructions. This is because the data set can be constructed from “three” of the four encrypted shares (as in the example above though any other specific number could be used), which is a “predefined number of the lowest amount of the plurality of data chunks” (thus meeting Google’s construction; EX1064, 13) and is “a threshold number of chunks less than or equal to the total number of data chunks” (thus meeting Patent Owner’s construction; EX1065, 9) from which the data set can be reconstructed after decryption.

⁹ To be clear, the “redundancy” Orsini refers to in this context is not with respect to individual share (*e.g.*, Orsini, [0282]-[0284] does not mean duplicating one or more shares), rather the redundancy refers to the fact that some portions of the data are distributed into multiple shares, rather than each share having non-overlapping portions of the data set.

4. [1B1] wherein generating the data chunks comprises:
distributing the data set into a plurality of shares, wherein
each of the shares comprises less than all of the data set,

103. As illustrated below in the colored portion of Orsini's Figure 21 I excerpted below, Orsini's "data units may be distributed into one or more **shares**" ([0116], [0296]), which meets both parties' proposed constructions because Orsini is "splitting the data set into a plurality of shares" (thus meeting Google's construction; EX1064, 2-5) and "dividing and placing the data set into a plurality of shares" (thus meeting Patent Owner's proposed construction; EX1065, 2). As also illustrated below, each **share** comprises a "portion[] of the original data" ([0118])—and thus each share "*comprises less than all of the data set,*" as claimed. See also Orsini (EX1005), [0117], [0281]-[0282], [0294]-[0296], claims, 1, 10, 20, 30.

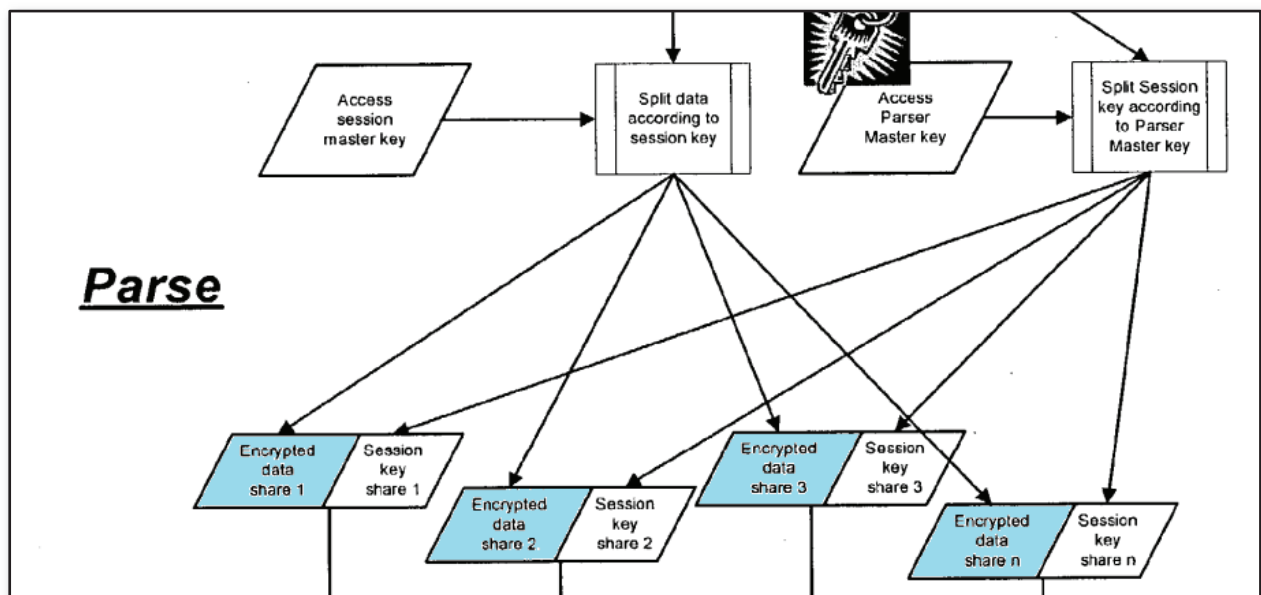


FIG. 21 (partial; color annotations added)

**5. [1B2] [wherein generating the data chunks comprises:]
accessing a plurality of distinct encryption keys,**

104. As I discuss below (§VII.B.6 [1B3]), Orsini’s parser program encrypts each share with a distinct share encryption key, and so Orsini’s parser program (in the cryptographic engine) is “*accessing a plurality of distinct encryption keys*,” as claimed. Orsini (EX1005), [0298]-[0299]; *see also* Orsini (EX1005), [0346]-[0352] (“multiple **distinct** secret key values: K1, K2, K3, Kn, K5” used to encrypt respective shares), [0395]-[0396] (explaining that Share Encryption Keys are “generated **to further encrypt the shares**”).

105. If limitation [1B2]’s “*accessing*” is understood to require that [1B2]’s “*distinct encryption keys*” must pre-exist, rather than be generated on-the-fly, POSAs would recognize the benefits in accesses the distinct encryption keys from a storage system, for the reasons I discussed above (§VII.B.2.b), where I explained why POSAs had reason to obtain encryption keys from a centralized key-management system.

**6. [1B3] [wherein generating the data chunks comprises:]
encrypting each of the shares with a respective one of the
plurality of distinct encryption keys**

106. As Orsini’s Figure 21 illustrates (a portion of which I’ve annotated below), Orsini’s parser program uses [1B2]’s individual **share encryption keys** (the claimed “**distinct encryption keys**”) to encrypt each **share** with a respective one of the **share encryption key**, thereby meeting limitation [1B3]. Orsini

(EX1005), [0298]-[0299]; *see also* Orsini (EX1005), [0346]-[0352] (“multiple **distinct** secret key values: K1, K2, K3, Kn, K5” used to encrypt respective shares), [0395]-[0396].

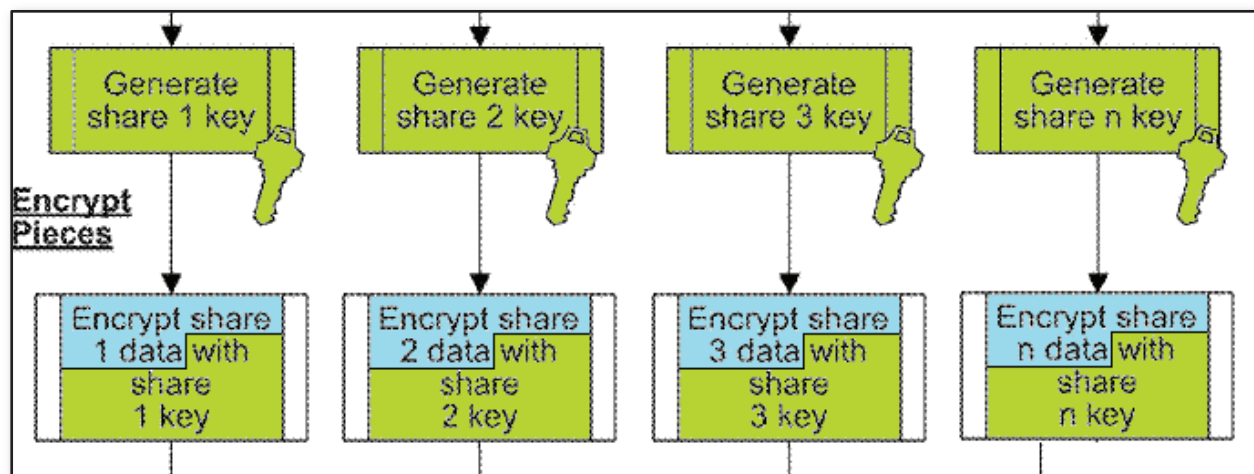


FIG. 21 (partial; color added)

7. [1C] performing an encryption operation based on the external key to further secure the plurality of data chunks; and

107. As I discuss below (§VII.B.7.a), Orsini discloses “*performing an encryption operation based on the external key to further secure the plurality of data chunks,*” as claimed. As I also discuss below (§VII.B.7.b), in view of Orsini’s disclosure, POSAs would have had reason to implement Orsini such that its encryption method included a step of “*performing an encryption operation based on the external key to further secure the plurality of data chunks,*” as claimed.

a. Orsini discloses [1C] (“performing an encryption operation based on the external key to further secure the plurality of data chunks”)

108. As I discuss below (§§VII.B.7.a.i-VII.B.7.a.iii), Orsini discloses “performing an encryption operation based on the external key to further secure the plurality of data chunks” in the three independent ways I discuss below.

i. Orsini discloses performing encryption steps “in different order”

109. *First*, in discussing Figure 21’s encryption method, Orsini teaches that “[i]t is readily apparent” to POSAs “that certain steps of the method[] described herein may be performed *in different order*.” Orsini, [0301]; *see also* Orsini, [0279] (noting “[t]he parsing process” is “designed in a modular tiered fashion,” which suggests and reinforces to POSAs that different steps of the encryption process could be reordered and/or duplicated using ordinary skill (because they are “modularized”)). Orsini thus discloses embodiments in which Figure 21’s four encryption phases – (1) encrypt with session master key, (2) parse, (3) encrypt with distinct share keys, and (4) obfuscate – are performed in different order.

110. POSAs understood that one “certain step[]” capable of being reordered was the first “encrypt” step with the session master key in Orsini’s Figure 21 (which I discussed above §VII.A) because it was well-known to reorder whether an encryption step is performed before and/or after splitting.

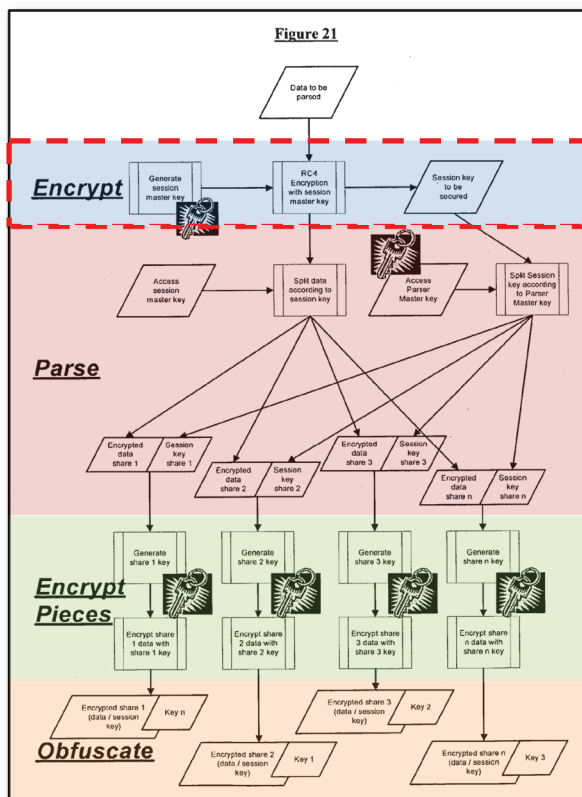
111. My opinion in this regard is corroborated by EX1028 (filed 2001) at [0090], which states that “employing a stream cipher after separation of the fragments into separate fragments is *logically equivalent to employing the stream cipher on the initial fragment stream* before its separation into separate fragment streams. In each case, the fragments are individually enciphered by the stream cipher process.” My opinion in this regard is further corroborated by EX1053 at [0026], which discloses a system “applies a division and encryption process” (where data is first “divided[]” “into a plurality of pieces and then encrypt[s] the pieces”) or “an encryption and division process” (where data is “first encrypt[ed]” “and then divid[es] the encrypted data into a plurality of pieces”). Likewise, EX1045 at Figures 11-13 and [0109]-[0113] discloses encrypting data before splitting (FIG. 11 and [0109], encrypting data after splitting (FIG. 12 and [0111]) and encrypting data before splitting and after splitting (FIG. 13 and [0113]), which corroborates my opinion that reordering an encryption step to occur after rather than before splitting a data set (or to occur both after and before splitting a data set) was conventional.

112. Thus, one different-order embodiment includes the session master key (an exemplary “*external key*,” see §VII.B.2.a.i above) being used to further encrypt data *after* it has first been parsed and encrypted with distinct share keys (that is, after the data set has been made into “*data chunks*”), rather than before either of

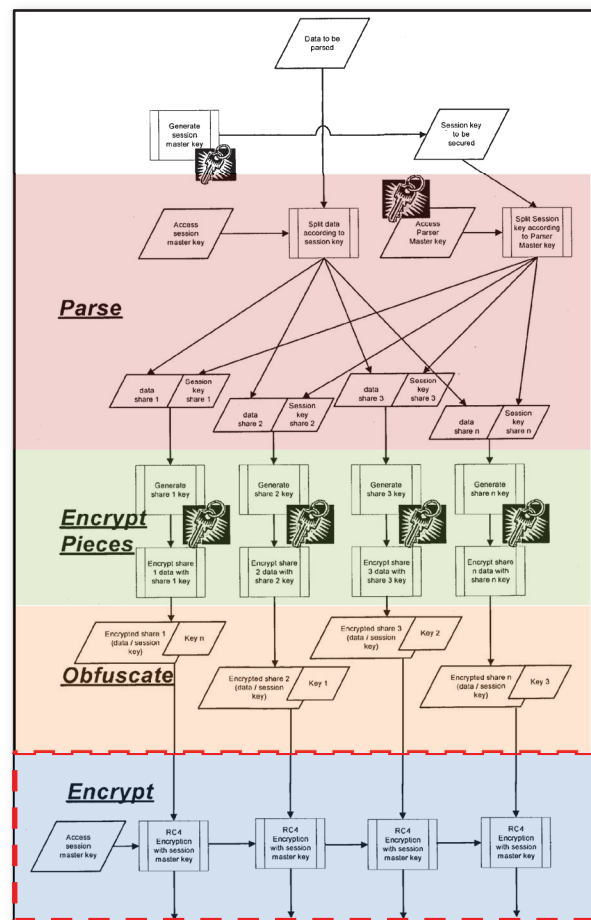
those steps. In this embodiment, each individual data chunk is “*further secure[d]*,” as claimed, because each data chunk undergoes a second encryption operation using the session master key (or second conventional session key), which meets the “*external key*,” for the reasons I discussed above §VII.B.2.

113. Two exemplary implementations of that embodiment – which are based on Orsini’s Figure 21 (left) – are conceptually illustrated below (right). In the first (illustrated below right), the encrypt step (blue) is reordered from being the first step in the process to the last step that occurs before storing the data:

FIG. 21 (original)

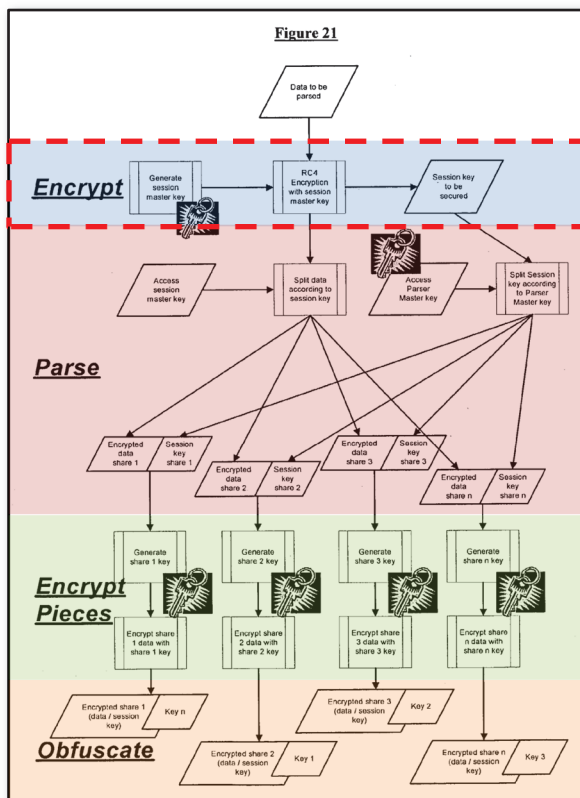


Different Order (Version 1)

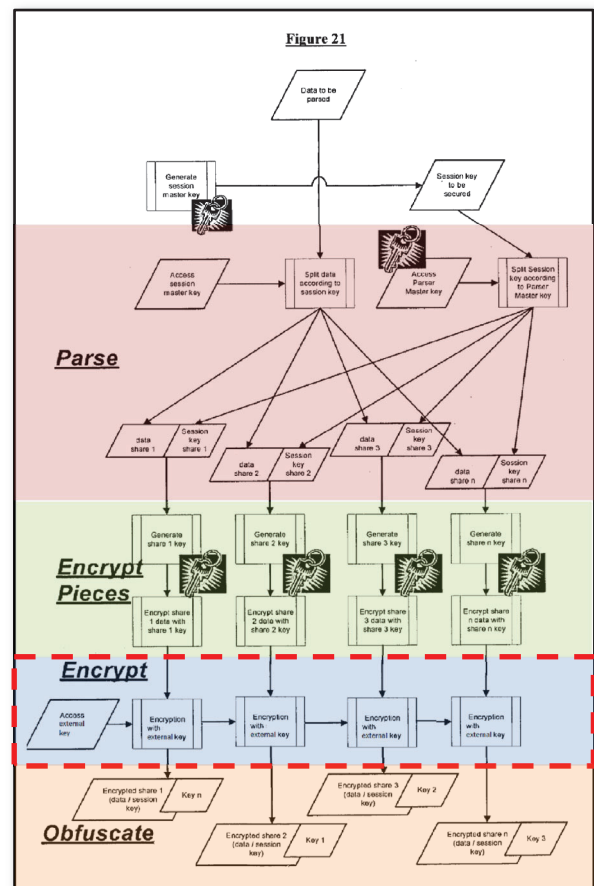


In the second exemplary implementation (illustrated below right), the encrypt step (blue) occurs after the data has been parsed and encrypted with distinct share-level encryption keys, but before the share-level encryption keys have been obfuscated (e.g., twice-encrypted share 1 is paired with share-key n; twice-encrypted share 2 is paired with share-key 1; etc.):

FIG. 21 (original)



Different Order (Version 2)



ii. Orsini discloses repeating encryption steps “multiple times.”

114. As the *second* way in which Orsini meets limitation [1C], Orsini teaches that the steps in its Figure 21 method can be “*repeated multiple times*.”

Orsini, [0301]. Orsini similarly has dependent claims that recite encrypting parsed data portions *a second time* using “a different encryption algorithm” than the algorithm that was used to encrypt the data portions the first time. *See, e.g.*, Claims 12 (claim 10’s encryption “step b”) which encrypts portions of data “is repeated using a different encryption algorithm”), 29 (claim 20’s encryption “step c”) which encrypts portions of data “is repeated one or more times” and “using an encryption algorithm that is different from...step a),” which “encrypt[s] a data set”), 39 (similar). That is, Orsini discloses that the data chunks (the parsed data portions that were encrypted a first time) are encrypted a second time (that is, the data chunks are further secured).

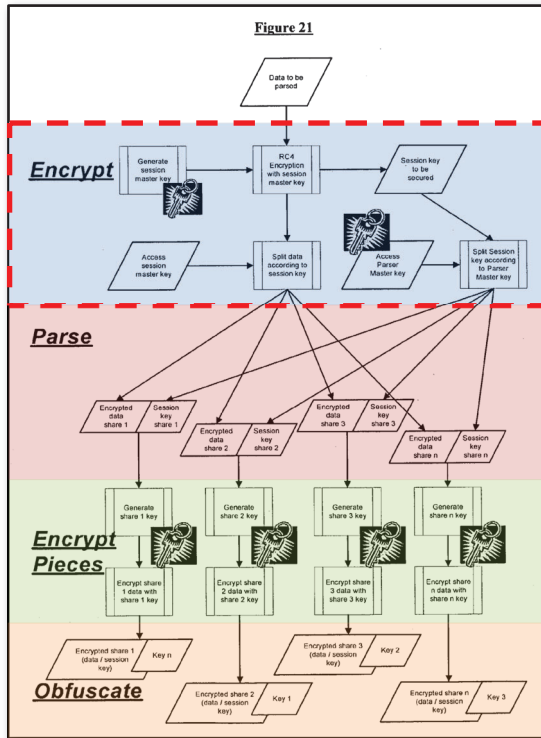
115. Orsini thus discloses embodiments where encrypted data shares (the “*data chunks*”) undergo an additional encryption step using an encryption algorithm that differs from the algorithm used to encrypt data shares the first time (which results in Orsini performing a second encryption after the encryption that came after the “parse” step).

116. One different algorithm performed after encrypting the “*shares*” (thus, after generating “*data chunks*”) that would have been obvious to POSAs includes using a single encryption key to further encrypt the data chunks with that key (*i.e.*, each share is encrypted (at least) twice), similar to (and providing comparable security to) using the session master key to encrypt the entire data set originally,

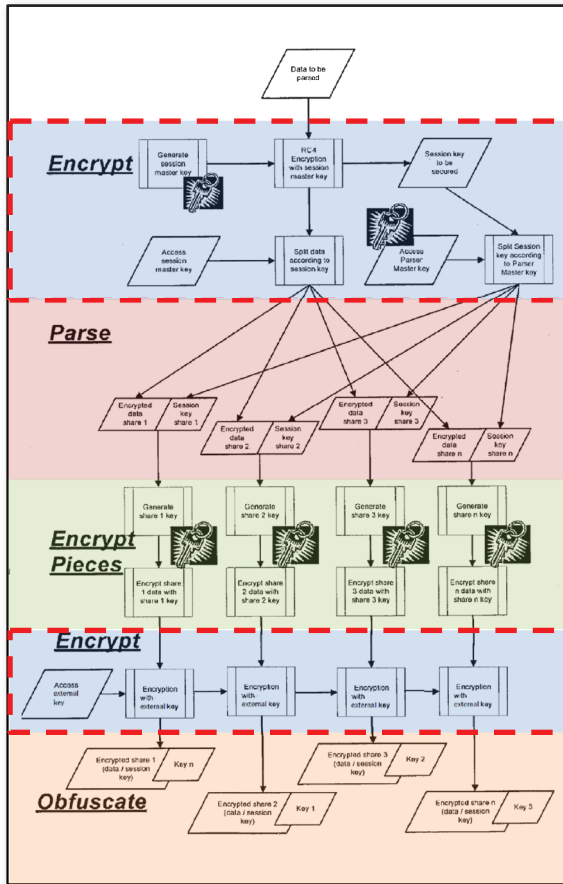
before splitting the data set. In other words, Figure 21’s method would have five steps, not four steps: (1) encrypt the original data set with a session master key, (2) parse the data into portions, (3) encrypt the data portions/shares with distinct share keys, (4) **encrypt each encrypted portion/share (the “*data chunk*”) with a unique key**, and (5) obfuscate the encrypted portions/shares. The common encryption key used to further encrypt the already-encrypted data chunks is, or had reason to be an “*external key*” for the reasons I discussed above (§VII.B.2).

117. An exemplary implementation of that two-encrypt-phase embodiment – which is based on Orsini’s Figure 21 (left) – is conceptually illustrated below. As shown below right, a first encrypt phase is initially applied to the data set and then a second encrypt phase is applied to the data chunks (that is, the second encrypt occurs after the shares have been encrypted with [1B2]’s “*distinct encryption keys*”):

FIG. 21 (original)

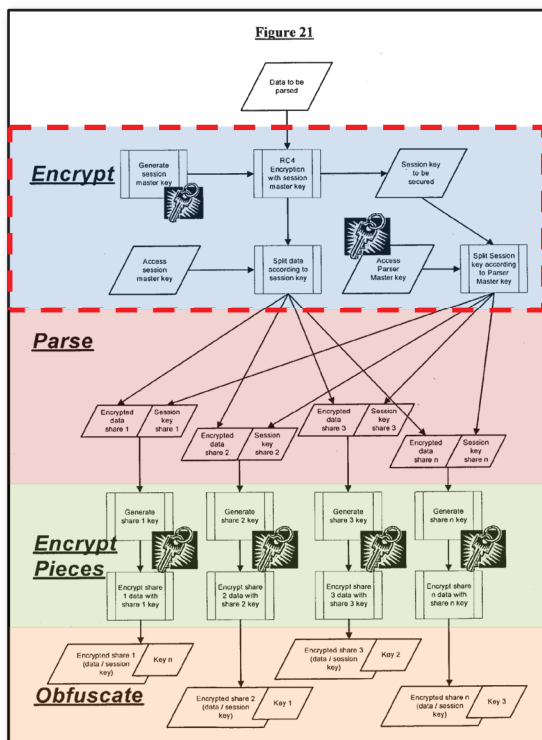


Two “Encrypt” Phases (Version 1)

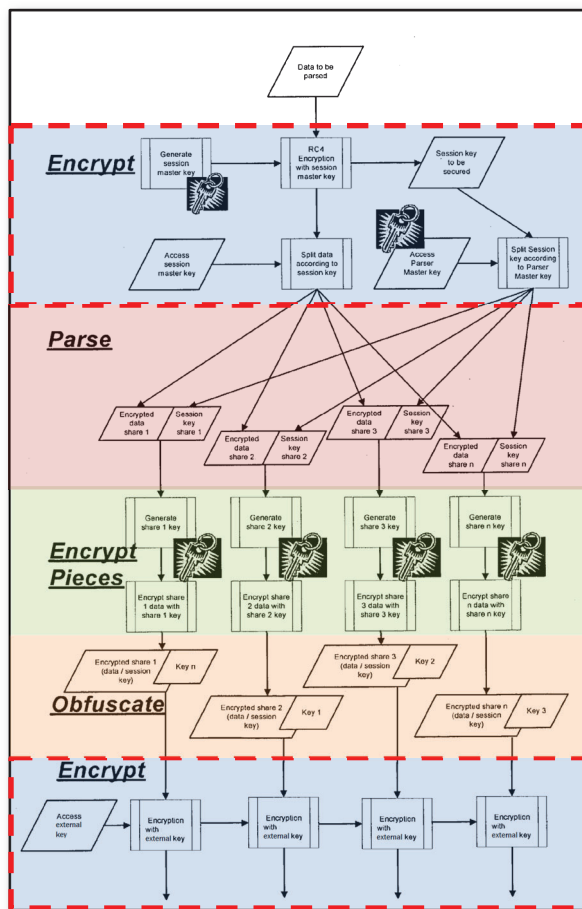


In a second exemplary implementation (illustrated below right), the second encrypt step occurs after (1) the data set has been split, (2) the shares have been encrypted with distinct share-level encryption keys, and (3) encryption keys have been obfuscated:

FIG. 21 (original)



Two “Encrypt” Phases (Version 2)



iii. Encrypting With a Public Key

118. As I discussed above (§VII.B.2.a.ii, [1A]), Orsini discloses using public encryption keys in its encryption techniques, which are additional “*external keys*” in Orsini under both parties’ proposed constructions, for the reasons I

discussed. Orsini discloses using the public encryption key(s) to further encrypt already-encrypted data (e.g., the encrypted shares, which are the “*data chunks*”). Orsini, [0097]. Orsini also discloses at [0107] that the “cryptographic handling module” in the “cryptographic engine” “may perform public-key encryption, symmetric-key encryption, or both.” “For example, public-key encryption may be used to *further secure the data* at the data storage facilities.” Orsini, [0119]. Orsini further teaches encrypting data with a public key before sending the encrypted data to storage by disclosing at [0095] that “incoming” “communication [to the depository] may advantageously be encrypted with public key encryption.” That is, Orsini further discloses an encryption step using the public encrypt key to further secure the data chunks by encrypting each data chunk (each encrypted share) with a public key before transmitting the data chunks to a depository for storage. (To the extent Patent Owner argues encryption with the public key is not performed or not necessarily performed by the cryptographic engine, Orsini is at least suggesting to a POSA that the cryptographic engine is performing that encryption and it would have been a conventional, obvious implementation because Orsini’s cryptographic engine performs the encryption process.) Encrypting with the public key(s) (“*external key(s)*”) is thus an encryption operation that “*further secure[s] the plurality of data chunks,*” as claimed.

119. Orsini also discloses “us[ing] the public key of the user” “to encrypt the session key that will be used to encrypt the document or message.” Orsini, [0172]. As I discussed above (§VII.B.6 [1B3]), an exemplary session key that Orsini uses to encrypt data is the session master key. Under the scope of the claims that Patent Owner attributes to [1C] in related district-court litigation, encrypting [1B3]’s “*distinct encryption keys*” with the “*external key*” meets [1C]. See EX1009, 8; §VII.B.7.b.iv below. If that interpretation were correct (which I have not been asked to consider), then encrypting Orsini’s session master key with a public key (an “*external key*”) would also meet [1C].

b. POSAs Had Reason To “*Perform[] an Encryption Operation Based on the External Key to Further Secure the Plurality of Data Chunks*”

120. Alternatively, POSAs had multiple reasons to implement Orsini to perform step [1C] in view of POSAs’ background knowledge, as I discuss below in §§VII.B.7.b.i-VII.B.7.b.iv.

i. POSAs had reason to re-encrypt data chunks with an external key

121. As I discussed above (§§VII.B.7.a.i-VII.B.7.a.ii), Orsini teaches “perform[ing]” “certain steps of the method” “*in different order*” or “*repeated multiple times*.” Orsini, [0301]. POSAs would have been motivated by those disclosures to reorder and/or repeat certain specific steps by re-encrypting “*data chunks*” directly with the same encryption key (e.g., the session master key or

another conventional encryption key), rather than encrypting the dataset before splitting data into the shares (which would be encrypted by [1B3]’s “*distinct encryption keys*” before all of the encrypted shares being encrypted again with the same external key). I illustrated examples of this re-encryption of data chunks above in §§VII.B.7.a.i-VII.B.7.a.ii. POSAs would have understood that encrypting the data chunks with the same encryption key would have provided various benefits. For example, beyond adding additional security to the data chunks by performing additional encryption operations, a POSA would have understood that encrypting multiple smaller chunks of data with the same encryption key would have been computationally easier than encrypting the single larger data set before splitting it into shares. This further encryption step would have also provided greater security by re-encrypting the data chunks a second time. *See also* §VII.B.7.b.iv below (discussing known benefits of encrypting encryption keys with so-called key-encryption-keys (“KEKs”), which Patent Owner contends meets the limitation, *see* EX1009, 8).

122. Thus, for the above reasons, it is my opinion that Orsini’s disclosure of “perform[ing]” “certain steps of the method” “*in different order*” or “*repeated multiple times*” ([0301]) would have also suggested and motivated to a POSA “*performing an encryption operation based on the external key to further secure the plurality of data chunks,*” as claimed.

123. POSAs would have also understood that encrypting all data after splitting with the same encryption key (that is, encrypting all data chunks with the same encryption key) would have been a known solution to the known problem of how to further secure data using encryption techniques, and thus would have been obvious to a POSA. My opinion in this regard is corroborated by EX1028 (filed 2001) at [0090], which states that “employing a stream cipher after separation of the fragments into separate fragments is *logically equivalent to employing the stream cipher on the initial fragment stream* before its separation into separate fragment streams. In each case, the fragments are individually enciphered by the stream cipher process.” My opinion in this regard is further corroborated by EX1053 at [0026], which discloses a system “applies a division and encryption process” (where data is first “divided[]” “into a plurality of pieces and then encrypt[s] the pieces”) or “an encryption and division process” (where data is “first encrypt[ed]” “and then divid[es] the encrypted data into a plurality of pieces”). Likewise, EX1045 at Figures 11-13 and [0109]-[0113] discloses encrypting data before splitting (FIG. 11 and [0109], encrypting data after splitting (FIG. 12 and [0111]) and encrypting data before splitting and after splitting (FIG. 13 and [0113]).

ii. POSAs had reason to re-encrypt data chunks with an externally-stored group key.

124. In my opinion, POSAs also had reason to “*perform[] an encryption operation based on the external key to further secure the plurality of data chunk*” in light of Orsini’s disclosure of embodiments in which “access to secured data can be limited to just those who are permitted or required to have access only to the portion they are working on, while others, such as managers or executives, may have access to all of the secured data.” Orsini (EX1005), [0412]. Encrypting with group-level keys was well-known before the alleged priority date of the ’854 patent and a conventional implementation of Orsini’s embodiments with group-based access would have been to re-encrypt data chunks with a group-level encryption key, thereby restricting access to data chunks based on a user’s membership in an authorized group(s). My opinion in this regard is corroborated, for example, by the following:

- EX1018 (filed 2004), 5:19-35: disclosing that a “data key data store 45 which contains data which can be used to find data keys assigned to users and **user** groups” and that data key data store 45 can beneficially “be set up centrally at a remote location” to allow “it to be isolated from the data processing facility” and to “allow[] particularly stringent security precautions” “such as particularly restrictive firewalls.”;

- EX1029 (filed 1999), 6:55-8:57: disclosing encrypting different portions of a document with different encryption keys to have multiple levels of access for different security and access levels, like different groups of individuals, similar to Orsini’s disclosure discussed above that only permits access to files based on membership in certain groups;
- EX1033 (filed 2003), 8:31-41: disclosing encrypting a file key with a “group key” so that only a “member of a group” allowed to access the encrypted file can decrypt it; and
- EX1034 (filed 2002), [0033]: disclosing “associat[ing] encryption keys with each group of users” and encrypting “portions of documents, 301, 302, 308, allowed to go to each group... with the corresponding group’s encryption key 306” such that “the resulting document contains different portions encrypted with different keys, 307.”

Thus, the above-cited prior art corroborates my opinion that POSA understood that a conventional way to achieve group-level access to data like that discussed in Orsini would have been to encrypt Orsini’s data chunks with a group-level encryption key.

125. POSAs also had reason for the group-level encryption key(s) to be received from an external storage system for the same reasons I discussed above

(§VII.B.2, [1A]) that POSAs had reason to receive any other encryption keys from an “*external storage system*.”

iii. POSAs had reason to reorder data within each chunk based on an external key.

126. In my opinion, POSAs also had reason to add additional security to Orsini’s encrypted shares (*i.e.*, the “*data chunks*”) by cryptographically reordering the data *within* each “*data chunk*,” which was a conventional cryptographic operation to secure data. This is expressly taught (or alternatively rendered obvious) by Orsini’s disclosure that the “distribution” of the data set’s data “could also involve a shuffling of the order of the units [of the data] in the shares.” Orsini, [0281]. (A “unit” of data could be, *e.g.*, a “byte” of data.) POSAs understood the benefits from shuffling data would also accrue if performed after encrypting each share with a respective encryption key (that is, shuffling the data within each “*data chunks*”). This is because in both cases the data being shuffled is further secured by the shuffling operation. And, as I discussed above (§VII.B.7.a.i), Orsini explicitly teaches performing encryption steps in different order.

127. Thus, Orsini’s disclosure of shuffling the data within shares and to perform steps in different order teach (or alternatively render obvious) shuffling the data within each “*data chunk*” (each encrypted share). My opinion is corroborated by, for example, EX1030 (filed 2004) at [0002], which notes as

background information that there are “complex ciphers [that] work” by “rearrang[ing] the data bits in digital signals.”¹⁰

128. In a conventional encryption technique that rearranges/reorders data, the reordering pattern (*e.g.*, the pattern that determines how bits/bytes of data are reordered) would be defined by an encryption key or a random/pseudorandom number generated based on the encryption key (*e.g.*, the key is a “seed” to a number generator). My opinion in this regard is corroborated, for example, by the following:

- EX1031, which illustrates in Figures 1-4 various techniques for rearranging portions of data (*e.g.*, bytes represented by letters) based on a pattern of 0s and 1s. EX1031 further explains at [0002]-[0003] that “there is occasionally a need to reorder some or all of the bits in one or more data elements according to a desired reordering pattern” ([0002]), which can be performed by a “computer program” ([0003]). EX1031 at [0016]-[0017] and [0021]-

¹⁰ EX1030 at [0002] refers to this as a “complex cipher” in comparison to “[s]imple ciphers” like those that merely “substitut[e] [] letters for numbers.” POSAs, however, would have understood that (at least) by the early-2000s, a cipher that rearranges digital data would not be “complex” in a conventional computer-based implementation.

[0022] describes examples of reordering based on a pattern as shown in Figures 1-2 and EX1031 at [0023] notes that these techniques are “efficient for applications requiring large numbers of data elements to be reordered,” include “application[s]” like “cryptography.”;

- EX1032 (filed 1998), which notes at 1:4-36 as background information that one form of encryption is “diffusion[, which] is introduced by spreading the plaint-text over a larger region in the cipher-text.” EX1032 illustrates an example of this “diffusion” in Figure 5, which shows a shuffle operation based on the pattern of a key “K.”; and
- EX1032, 4:3-26 (describing a “shuffling operation [that] creates diffusion in the data by moving the bits around), 6:23-7:16 (describing how “encryption [with] the shuffle operation” can come after an “XOR operation” (another encryption step) and that the shuffling is done “with the help of the key K”).

129. My opinion that it was conventional to use an encryption key to form a random/pseudorandom number for use in a reordering operation is corroborated by, for example, EX1040 (filed 2004) at [0030], which discloses an “encryption key EK is fed as a seed to the pseudo random number generator” and by EX1041 (filed 2004) at [0024], which similarly discloses that “[t]ypically, the initialization of the random number generator, that is the so-called seed, will be the secret or the session key” that is used “to perform encryption.”

130. Thus, for the above reasons, in my opinion, POSAs had reason to further secure Orsini’s encrypted shares (the “*data chunks*”) based on one of the external keys I discussed above (§VII.B.2, *e.g.*, Orsini’s session master key or another conventional encryption key), as claimed, by having that external key define the reordering pattern within each “*data chunk*” or by having that external key act as a “seed” to generate the reordering pattern (*e.g.*, a random/pseudorandom number) used to define the reordering.

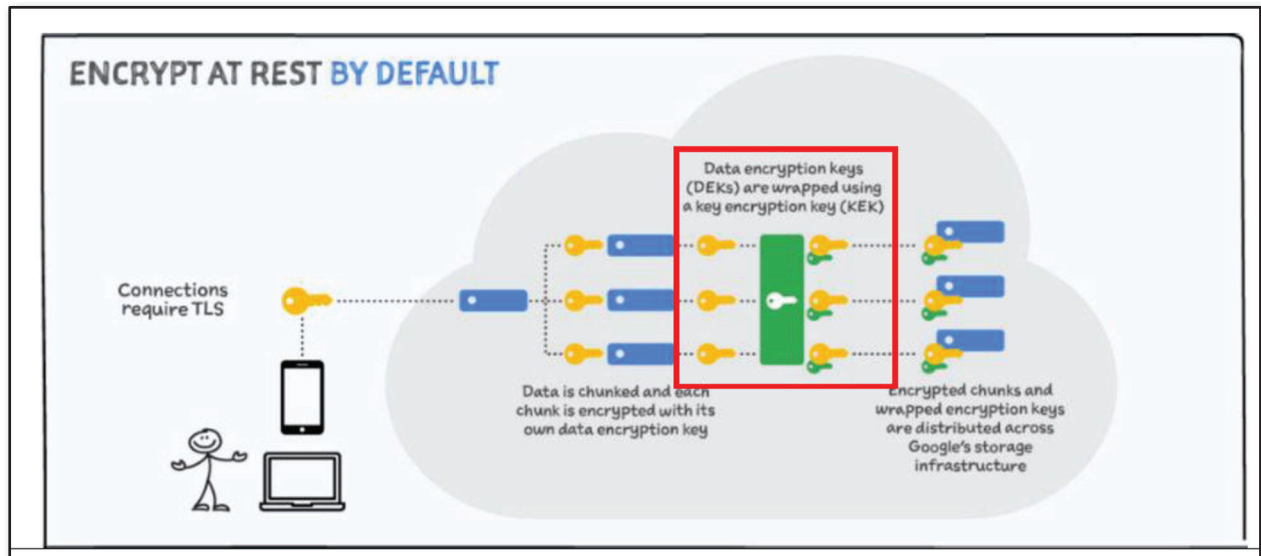
iv. POSAs had reason to encrypt [1B3]’s keys with an externally-stored key-encryption key.

131. POSAs also had reason to encrypt each of [1B3]’s “*distinct encryption keys*” (which map to Orsini’s share encryption keys as I discussed in §VII.B.6 above) with a common encryption key, which was (and still is) known in the art as a key-encrypting key (“KEK”). This is corroborated, for example, by EX1035 (published 1980) at 8:15-22, which describes “two categories of cipher keys” for use in a “cryptographic system”: “operational keys (KO) and key encrypting keys (KEK).” As discussed above (§VII.B.7.a.iii, [1C]), Orsini itself discloses (or alternatively renders obvious) encrypting session keys with a public-key encryption key. Orsini, [0172].

132. As I discuss below (§VII.B.8, [1D]), Orsini stores [1B3]’s distinct encryption keys (with “*data chunks*”) on storage systems. However, as EX1035 at 10:42-62 corroborates, POSAs knew, in general, “cryptographic keys should never

occur in clear form outside the cryptographic device.” EX1035, 10:42-62 (noting that “cryptographic keys should never occur in clear form outside the cryptographic device, except under securing conditions during the period when keys are originally distributed and installed or when stored in a secure place such as a safe, vault or similar location for backup or recovery”; accomplishing this by “encipher[ing] the terminal master key”). Thus, POSAs had reason to encrypt Orsini’s distinct encryption keys with a KEK. The known, beneficial use of a KEK is further corroborated by EX1036 (filed 2000) at 1:66-2:6, which discloses as background information that “[s]ecurity for such a system, where the encryption keys are included with the encrypted message [like in Orsini], is not to store or transmit the encryption keys in an easily readable format but rather to encrypt the encryption keys themselves. This approach means that the **data encryptions keys will be encrypted by a key encryption key.**” Thus, alternative or in addition to encrypting session keys with a KEK, POSAs had reason to encrypt [1B3]’s keys with a KEK, to avoid a potential security risk in storing [1B3]’s encryption keys in clear (which is an unencrypted) form on storage devices (especially those accessible from the public Internet), POSAs had reason to encrypt [1B3]’s keys with a KEK.

133. As shown below, under the scope of the claims that Patent Owner attributes to [1C] in its district-court infringement contentions, encrypting [1B3]’s keys with an (allegedly) externally-received KEK meets [1C]. See EX1009, 8.



EX1009, 9: Patent Owner’s District Court Infringement Contentions

POSAs had reason for a KEK in Orsini’s system to be retrieved from an “*external storage system*” for the same reasons POSAs had to implement Orsini to receive encryption keys (*e.g.*, its session master key) from an external storage system, as I discussed above (§VII.B.2.b, [1A]).

* * *

134. In my opinion, POSAs would have had a reasonable expectation of success in any of the conventional encryption operations that I discussed above (*e.g.*, repeating or reordering individual encryption steps, reordering data within chunks, group-level encryption keys, public-key encryption, and/or encrypting

[1B3]’s encryption keys with a KEK), because implementations of those operations were conventional and well-known to POSAs and Orsini itself discloses that the “[e]ncryption of data may be performed prior to or after the first parsing, splitting or separating of the data” and “the encrypting set may be repeated for one or more portions of the data” ([0013]). Orsini (EX1005) further discloses that “a variety of combination of [encryption method] steps may be desired” ([0230]), which would have reinforced a POSA’s reasonable expectation of success in using any of the encryption methods I discussed above in combination with Orsini’s system.

8. [1D] storing with the plurality of data chunks data indicative of at least one of the distinct encryption keys on a plurality of different storage devices.

135. I have compared Orsini’s disclosure to the ’854 patent’s disclosure (*see* EX1010) and, with respect to the distinct encryption keys used to encrypt each share, Orsini’s disclosure is substantively identical to the ’854 patent’s. My opinion is corroborated by the portions of EX1001 that Patent Owner cites as intrinsic evidence that allegedly supports its construction of the “data indicative” term. *See* EX1065, 4 (citing EX1001, 31:39-45, 48:33-43, and 74:62-66). EX1001 at 31:39-45 and 48:33-43 are identically disclosed in Orsini at [0162] and [0247], respectively. EX1001 at 74:62-66 discloses using a “hash function” with “random or pseudo-random value” to “indicate to which share to append” a

“respective bit” of data. Orsini similarly discloses “perform[ing] hashing” and “hash comparisons” (Orsini, [0134]) and “partition[ing] the data” “based on...random” values (Orsini, [0280]-[0281]).

136. Thus, Orsini discloses “*storing with the plurality of data chunks [i.e., encrypted shares] data indicative of at least one of the distinct encryption keys on a plurality of different storage devices*” to the same extent the ’854 patent discloses this limitation.

137. As Figure 21 illustrates (a portion of which I color-shaded below), Orsini stores **encrypted shares** (which meet the claimed **data chunks**, *see* §VII.B.6 above) with a share encryption key (which meets one of [1B3]’s “**distinct encryption keys**,” *see* §VII.B.6). The share encryption key is “*data indicative*” of that key under Patent Owner’s construction, which includes an encryption key itself. EX1065, 3-4 (data that “makes known” or “shows” the key). My understanding that Patent Owner’s construction includes a key itself is corroborated by the claim scope Patent Owner attributes to [1D] in related district court litigation, where Patent Owner alleges the limitation is by “by virtue of storing DEKs” – data encrypting keys that are alleged to meet [1B] – “with the data chunks.” EX1009, 9 (alleging limitation is met because “chunks of data are stored with their own DEK and are ‘stored across multiple server’”).

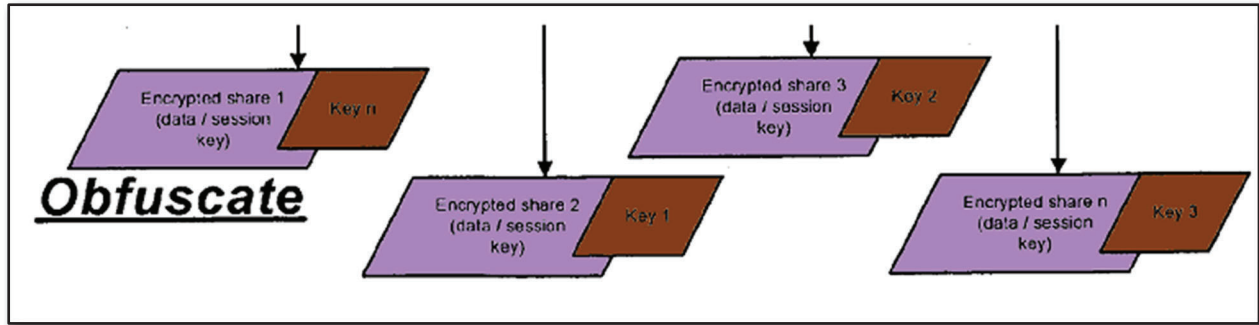


FIG. 21 (partial; annotated)

The share encryption keys used to encrypt shares are stored “in different locations from the encrypted data portions or shares” (e.g., “Share 1 gets Key 4” etc.).

Orsini (EX1005), [0299]. These different locations store the **encrypted share** (a “data chunk”) and **share encryption key** (a “distinct encryption key”) “on one or more data depositories or **storage devices**” (and thus “on a plurality of different storage devices,” as claimed). Orsini (EX1005), [0379]; see also Orsini (EX1005), [0018], [0111]-[0112], [0154]-[0155], FIGs. 7, 21, Claims 10, 13-16.

138. POSAs also had reason to store with each encrypted share “data indicative of” a share encryption key under Google’s construction of “data indicative of ... key(s)” as “metadata about ... key(s); not an encryption key.” EX1064, 5-7. As I discussed above (§VII.A), Orsini supported multiple prior-art encryption algorithms, like AES. Orsini, [0378]. POSAs understood that AES (and other conventional encryption algorithms) supported multiple so-called encryption modes. My opinion in this regard is corroborated by EX1039, which discloses at 2:58-3:11 that encryption algorithms like DES, triple DES, and AES

supported “various mode...to implement particular encryption or decryption functions.” These “modes include, for example, cipher block chaining mode (CBC), counter mode (CTR) and output feed back mode (OFB).” Thus, in order to indicate which encryption mode was used, POSAs had reason to store each encrypted share with data indicating the encryption mode associated with a share encryption key, “metadata about” the encryption key, and thus meets Google’s construction of “*data indicative of...[a] distinct encryption key.*” EX1064, 5-6.

139. POSAs also understood that conventional encryption algorithms supported variable-sized encryption keys, to provide additional security. For example, EX1039 at 2:58-67 disclosed that common encryption algorithms, like the AES standard mentioned in Orsini ([0378]), allowed for “variable size messages.” POSAs also knew that AES, for example, had various keys sizes (*e.g.*, 128-, 192-, or 256-bit length keys). *See* EX1066, [0009] (“AES ... uses cryptographic keys of 128-, 192-, and 256-bits.”). Similarly, EX1056 at 3:8-24 disclosed that for sub-encryption keys (keys encryption portions of a data set) “it is preferred for the sake of maximizing security that they be of variable length.” When using variable sized encryption, POSAs had reason to store each share with data indicating the length of the share encryption key stored with the share, which is metadata about the key, and thus also meets Google’s proposed construction of “*data indicative of*” a “*distinct encryption key.*” EX1064, 5-6.

140. Thus, in my opinion, Orsini discloses and renders obvious limitation [1D] under both parties' proposed constructions.

141. If [1D] were alternatively interpreted to further require that *each* “data chunk” is stored “on a plurality of different storage devices,” Orsini meets this as well. This is because Orsini teaches that the system “can be enhanced” to provide robustness to “data recovery issues” by “provid[ing] *a mirror of the...shares* and session master key depository” *e.g.*, using “*replication*” (or RAID storage), which POSAs understood duplicates each chunk and stores each chunk’s copies on more than one storage device. Orsini, [0388].

142. As I discussed above (§§VII.B.2-VII.B.8), Orsini performs limitations [1A]-[1D] in the order in which claim 1 recites those steps. To the extent Patent Owner argues any of the steps are performed in a different order, performing the steps in the recited order would have been obvious as a conventional design choice, that would have required only ordinary skill for a POSA to implement with a reasonable expectation of success. (Indeed, as I noted above, §VII.B.7.a.i, Orsini discloses and claims encryption steps being performed in various orders.) Orsini thus meets Google’s construction requiring claim 1’s steps to be performed in order. EX1064, 16.

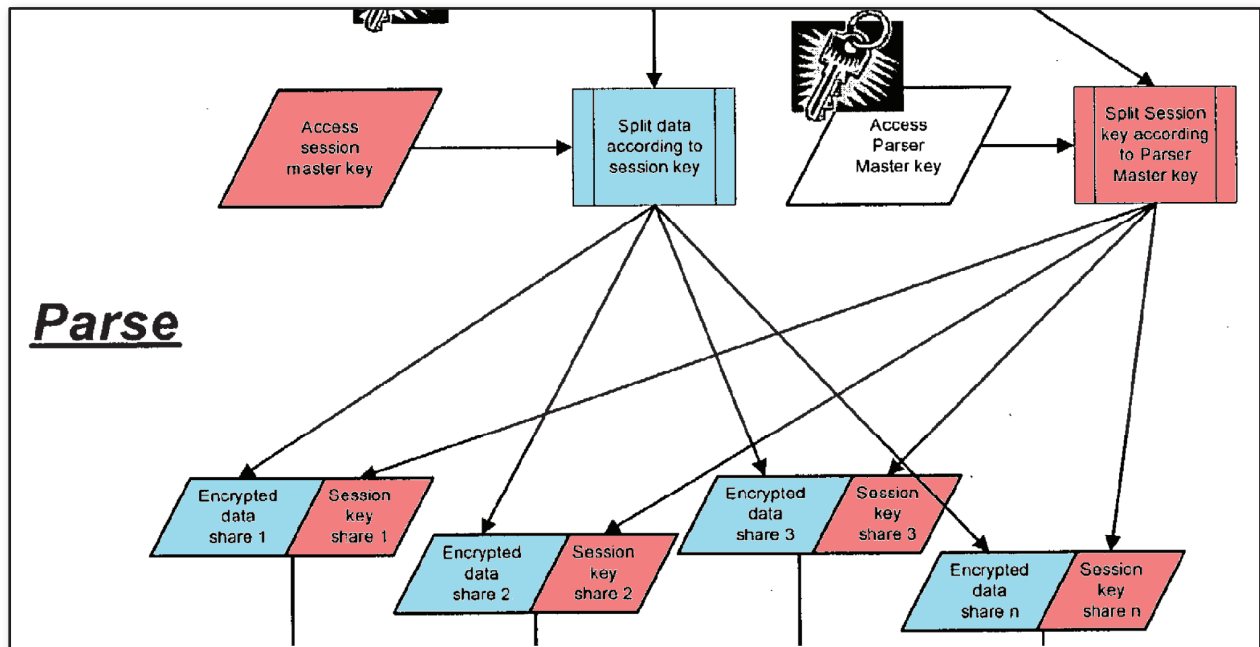
C. Claim 2: The method of claim 1, wherein storing with the plurality of data chunks data indicative of at least one of the distinct encryption keys comprises storing each of the plurality of data chunks with at least one of the distinct encryption keys.

143. As I discussed above (§VII.B.8, [1D]), each of Orsini’s encrypted shares (which maps to the “*data chunks*,” §VII.B.3[1B]) is stored with one of the share encryption keys (which map to the “*distinct encryption keys*,” §VII.B.6)—specifically a key that was used to encrypt a different one of the shares. Orsini, FIG. 21, [0299] (“Encrypting each share, then store the encryption keys in different locations from the encrypted data portions or shares: Share 1 gets Key 4, Share 2 gets Key 1, Share 3 gets Key 2, Share 4 gets Key 3.”). This meets claim 2 under the claim scope Patent Owner attributes to claim 2 in related district court litigation, where Patent Owner alleged claim 2 is met by “storing a DEK [data encryption key] with each data chunk.” EX1009, 10-11.

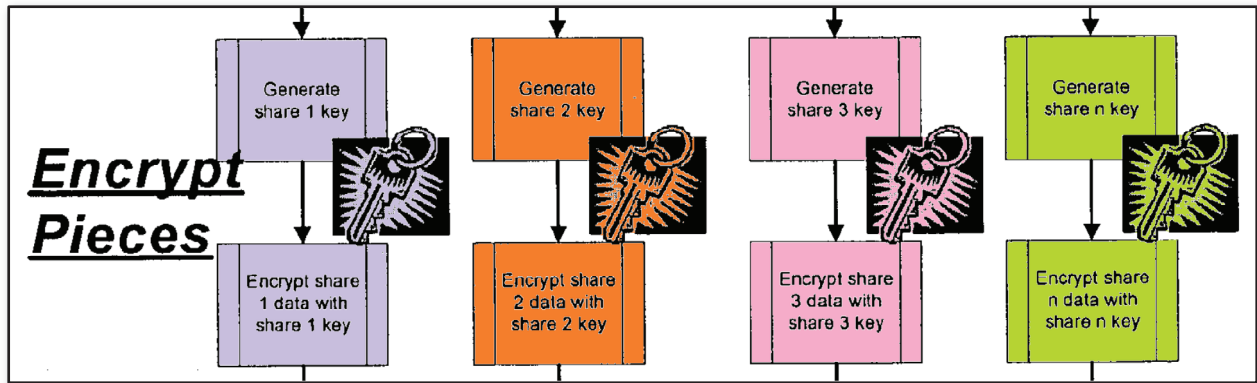
D. Claim 3: The method of claim 1, wherein storing each of the plurality of data chunks with at least one of the distinct encryption keys comprises including with a particular data chunk of the plurality of data chunks data indicative of an encryption key that was used to encrypt a different data chunk of the plurality of data chunks.

144. Orsini meets claim 3 for the same reason it meets [1D] (which I discussed above in §VII.B.8) and claim 2 (which I discussed above in §VII.C). For example, as illustrated in Orsini’s Figure 21 (portions of which I excerpted and

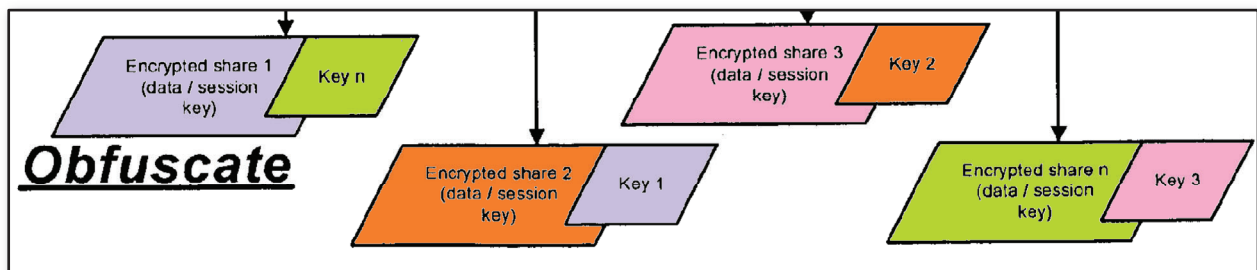
annotated below), Orsini discloses paired share, which are illustrated as the blue-red combinations below:



Orsini discloses that each paired share above is encrypted with its own distinct encryption “share key” (purple, orange, pink, and green below), resulting in an encrypted share that includes a portion of the (encrypted) original data set and a portion of the original session master key. Orsini (EX1005), [0298]-[0299], [0395]-[0396]. This is depicted below with an colored-shaded excerpt from Orsini’s Figure 21:



145. Orsini further discloses that each encrypted share is paired with one of the share keys used to encrypt a *different* share (*e.g.*, encrypted share 2 is paired with share 1 key, etc.). Orsini (EX1005), [0299]. This is depicted below with an annotated excerpt from Orsini’s Figure 21:



Each pair of encrypted share and share key can then be stored, for example, in different locations (*e.g.*, data depositories), including different physical locations. Orsini (EX1005), [0299], [0379]-[0380].

E. Claim 4: The method of claim 1, wherein distributing the data set into a plurality of shares comprises using a random technique or pseudorandom technique.

146. As discussed above §VII.B.4 ([1B1]), Orsini’s exemplary method in Figure 21 distributes the data set into a plurality of shares based on the session

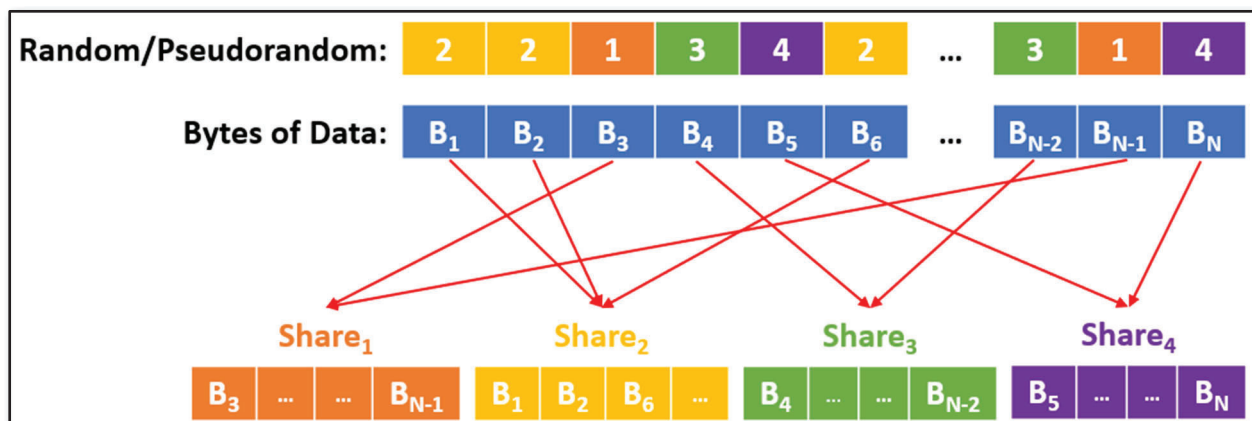
master key. Orsini, [0087], [0280], [0393]-[0394]. As I discuss below, Orsini discloses several different ways to perform that step, including random or pseudorandom techniques.

147. Orsini discloses a “cryptographic split process” that distributes the data set into, for example, “4 *shares*” based on a “*random* distribution” of numbers “corresponding to the four shares.” Orsini (EX1005), [0282]; *see also* §VII.B.4 ([1B1], above). That is using a random technique in distributing the data set into shares. Thus, Orsini discloses “*distributing the data set into a plurality of shares comprises using a random technique,*” as claimed. *See also* Orsini (EX1005), [0105], [0116], [0124], [0281], [0284], Claim 54.

148. Orsini also discloses that data can be distributed “into ‘n’ segments, or shares, according to the value of K5,” which is a “secret key value[.]” Orsini (EX1005), [0346]-[0348]. This “results in ‘n’ *pseudorandom* shares of the original encrypted data” ([0348]), and thus is a pseudorandom technique for distributing the data set into shares. Thus, Orsini also discloses “*distributing the data set into a plurality of shares comprises using a ... pseudorandom technique,*” as claimed.

149. Below I have illustrated an example of distributing the data set into shares using a random or pseudorandom technique consistent with Orsini’s disclosure, where a random or pseudorandom sequence of numbers defines what

share a unit of the data set is distributed to. For example, as illustrated below, the first number in the sequence is a “2” and so the first unit of data (e.g., a byte of data) is distributed to share 2 when generating the shares:



F. **Claim 5: The method of claim 1, wherein distributing the data set into a plurality of shares comprises using a deterministic technique or pseudorandom technique.**

150. Orsini meets claim 5 for the same reason Orsini meets distributing the data set into a plurality of shares using a pseudorandom technique, which I discussed above (§VII.E, claim 4).

151. Orsini also discloses distributing the data set into a plurality of shares using a deterministic technique, e.g., by using “predetermined” (e.g., “same”) sized portions of the data set. Orsini (EX1005), [0116]. Orsini at [0281] similarly discloses “cryptographic split (cryptosplit) partition[ing of] the data into N number of shares,” using “any pattern or combination of data unit sizes whether predetermined or randomly generated.” See Orsini (EX1005), [0281] (“The units can also be of different sized, *based on either a random or predetermined set of*

values.”). Thus, Orsini also meets “*wherein distributing the data set into a plurality of shares comprises using a deterministic technique,*” as claimed.

G. Claim 6: The method of claim 1, wherein each share of the plurality of shares comprises a substantially random distribution of a subset of the data set.

152. Orsini meets claim 6 for the same reason Orsini meets distributing the data set into a plurality of shares using a random or pseudorandom technique (which is a substantially random technique for distributing a subset of the data set), which I discussed above (§VII.E, claim 4). *See, e.g.,* Orsini (EX1005), [0281], [0346]-[0348].

H. Claim 7: The method of claim 1, wherein performing an encryption operation based on the external key to further secure the plurality of data chunks comprises encrypting primary data with the external key.

153. As I discussed above (§VII.B.7.a.i, [1C]), Orsini discloses “us[ing]” the session master key (the “*external key*” that I discussed above in §VII.B.2.a.i and §VII.B.7.a) “to encrypt the data prior to the parsing operation.” Orsini (EX1005), [0393]-[0394]. Claim 7 recites “*primary data*” without antecedent basis for the “*primary data*,” and the ’854 patent’s specification does not explain what “*primary data*” means. If Claim 7’s “*primary data*” is understood to include the “*data set*” *before* it’s distributed into shares (*see* §VII.B.4), Orsini meets “*encrypting primary data with the external key*” (*e.g.,* Orsini’s session master key), under that understanding, because as I discussed above (§VII.B.7.a.ii, §VII.B.7.b.i)

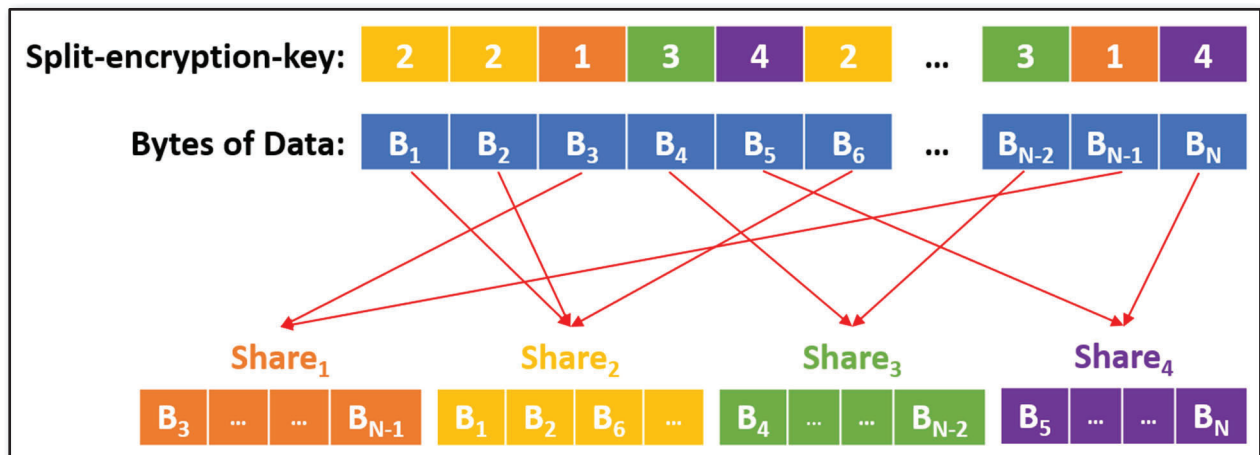
Orsini discloses or alternatively renders obvious embodiments that repeat an encryption step (*e.g.*, encrypts with the session master key both before *and* after encrypting each share with a share encryption key). Thus, Orsini meets encrypting “primary data with the external key,” as claimed.

154. And if Patent Owner argues that the limitation Claim 7 adds to Claim 1 can be met by encrypting [1B3]’s keys with a key-encryption-key (KEK) received from an external storage system, POSAs had reason to do so for the reasons I discussed above (§VII.B.7.b.iv, [1C]).

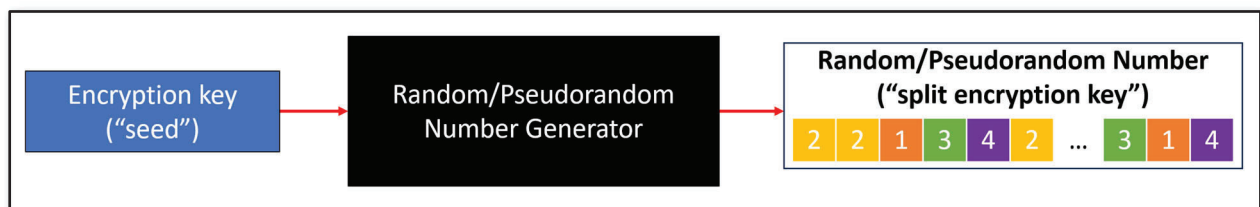
I. Claim 8: The method of claim 1, wherein distributing the data set into a plurality of shares comprises: generating a split encryption key based on the external key; and distributing the data set based on the split encryption key.

155. As I discussed above (§VII.E, claim 4), Orsini discloses distributing the data set into the plurality of shares based on a random or pseudorandom technique. For example, Orsini discloses “*a key* would be obtained to create a sequence of 23 random numbers ... each with a value between 1 and 4 corresponding to the four shares.” Orsini (EX1005), [0282]. I’ve illustrated below an exemplary “key” that provides a number sequence for splitting data into shares based on Orsini’s disclosure. The ’854 patent (EX1001) (at 78:11-28) identifies a “split encryption key” as a key used in the “split algorithm” that “split[s data] into a predefined number of shares.” Thus, the exemplary number sequence I illustrate below based on Orsini’s disclosure at [0282] (which, as discussed above, discloses

“a **key**” “obtained to create a sequence of 23 random numbers”) is a “*split encryption key*” or is a sequence based on the split encryption key, because it is the sequence/key used in the algorithm for distributing data into shares.



156. As I noted above, Orsini says the random/pseudorandom number (“*split encryption key*”) is itself created using a key. Orsini (EX1005), [0282]. This is consistent with a conventional random/pseudorandom number generator (which I have illustrated below), which takes as its input an encryption key as a so-called “seed” and outputs a random/pseudorandom number:



Alternatively, the split encryption key could itself be fed into the random/pseudorandom number generator to generate a random sequence for

distributing data. *See* EX1054, [0005] (disclosing as background information that “[e]ncryption keys are random strings of numbers”).

157. My opinion in this regard is corroborated by EX1040 at [0030], which discloses using an encryption key “as a seed to” a “pseudo random number generator 6 that outputs a random number.” My opinion is further corroborated by EX1041 at [0024], which discloses that “[t]ypically, the initialization of the random number generator, that is the so-called seed, will be the ... session key”) at discloses at [0026]-[0027] that “output sequence” of the number generator is a “key sequence,” which is consistent with Orsini’s disclosure (at [0282]) that a random sequence of numbers used to distribute data into the shares is a “key.”

158. As I discussed above (§VII.E, §VII.F), Orsini disclose distributing the data set into the plurality of shares based on the session master key (the claimed “*external key*,” as I discussed above in §VII.B.2.a.i and §VII.B.7.a). Orsini (EX1005), [0296], FIG. 21. Thus, in Orsini’s embodiments that use a random/pseudorandom technique to distribute data (*e.g.*, Orsini (EX1005), [0282]), POSAs would understand that the session master key is the key used to generate the random/pseudorandom number that is used as the “*split encryption key*.”

159. But even if Orsini were generating the split encryption key in another way, POSAs would have had reason to generate that split encryption key using the session master key because it would have been a simple substitution of one

technique for generating a random/pseudorandom number (whatever technique Orsini uses to generate the split-encryption-key) for another (using the session master key as a number-generator's "seed"), which, as I discussed above, would have been the predictable (and conventional) result of generating random/pseudorandom numbers based on an encryption key.

160. Thus, Orsini discloses, or alternatively renders obvious, using a "*split encryption key*" (e.g., a pseudorandom/random number sequence that is itself, or is generated from, the split encryption key) that is based on the session master key or implemented as an external key, and Orsini discloses and renders obvious distributing the data set into a plurality of shares based on that split encryption key (or a number sequence generated from the split encryption key), which is (or would have been) based on the external key, as claimed.

J. Claim 9: The method of claim 8, further comprising encrypting each of the shares with the external key.

161. As I discussed above (§VII.J), Orsini discloses and renders obvious claim 8. As I discussed above (§VII.B.7.a.i, [1C]), Orsini discloses and POSAs had reason to reverse the order of encryption with the session master key (an exemplary "*external key*"), by encrypting the data chunks with that key, which meets Claim 9 because each of the shares within each data chunk would be encrypted by the ("*external*") session master key.

162. For example, as discussed above, Orsini discloses (§VII.B.7.a.i) and POSAs had reason (§VII.B.7.b.i) to reverse the order of encryption with the session master key, by encrypting the data chunks with the session master key. As another example that I discussed above, Orsini discloses (§VII.B.7.a.ii) and POSAs had reason (§VII.B.7.b.i) to repeat the parsing of data with the session master key. Either of those embodiments encrypt the shares with the (“*external*”) session master key, as claimed.

163. Similarly, as I discussed above (§VII.B.7.a.iii, §VII.B.7.b.i), Orsini discloses and renders obvious encrypting shares or chunks with a public-encryption key, which is another exemplary external key disclosed in Orsini (§VII.B.2.a.ii, [1A]). Orsini, [0072], [0092]. Encrypting shares with the public encryption key is thus another way Orsini meets Claim 9.

164. Finally, if Patent Owner argues that encrypting [1B2]’s encryption keys with a key-encryption key (KEK) meets claim 9, it would have also been obvious to encrypt [1B2]’s distinct encryption keys with a KEK received from external storage, for the reasons I discussed above (§VII.B.7.b.iv, [1C]).

K. Claim 10: The method of claim 9, wherein the data set is recoverable using at least a minimum number of the plurality of data chunks and the external key.

165. Orsini meets claim 10 for the same reason I discussed above that Orsini meets [1B1] (§VII.B.4) – which recites that the data set can be reconstructed

using at least a minimum number of the plurality of data chunks – and meets claim 9 (§VII.J) – which recites encrypting each of the shares with the external key. Orsini at [0300] discloses that “[t]o restore the original data format, the steps are reversed.” Thus, Orsini’s data set can be recovered using at least a minimum number of shares and the encryption key, as claimed.

VIII. GROUND 2: CLAIMS 1-10 WOULD HAVE BEEN OBVIOUS OVER FOSTER-HAYHURST

166. I have been asked to provide my opinion concerning whether claims 1-10 of the '854 patent would have been obvious to a POSA in light of the disclosures of Foster, Hayhurst, and the knowledge of a POSA. For the reasons explained below (§§VIII.D-VIII.M), it is my opinion that each of claims 1-10 would have been obvious to a POSA over the combination of Foster and Hayhurst.

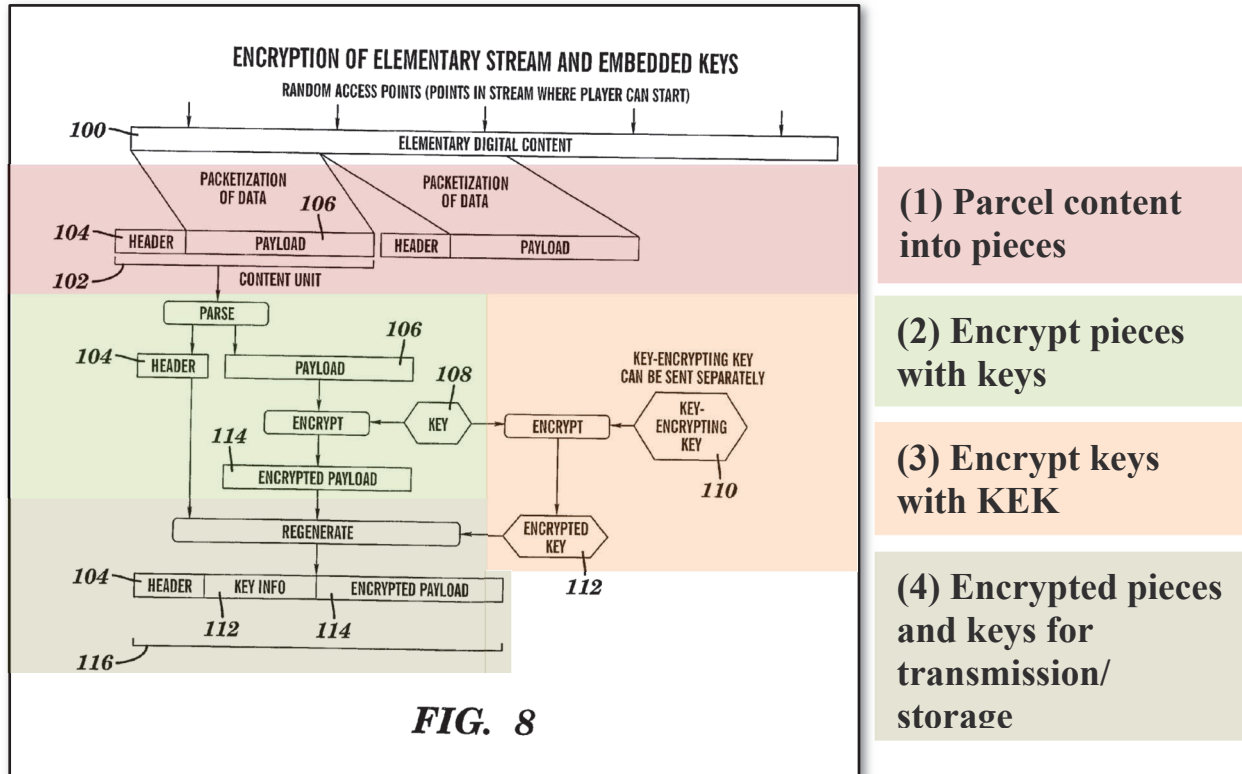
A. Foster (EX1006)

167. Foster discloses methods for securing digital content, like a data set for a digital movie, using encryption techniques that “attach[] a title [encryption] key to encrypted content for synchronized transmission to, or storage by, a recipient.” Foster (EX1006), Abstract, [0002]. Foster’s digital content includes “*any* data such as digital image, sound or binary data.” Foster (EX1006), [0037].

168. Foster teaches that prior-art encryption methods used “multiple keys” to “provide[] enhanced security by preventing an entire content stream from being accessed with a single key.” Foster (EX1006), [0004]. Those techniques, however, typically stored encryption keys separately from the encrypted content portions, and therefore required subsequent “synchronization” of each encrypted portions with its corresponding encryption keys and also created challenges in “provid[ing] random access” to portions of content. Foster (EX1006), [0006].

169. To address these and other deficiencies in the prior art, Foster's encryption methods involve (1) parceling (splitting) content up into multiple content packets, (2) encrypting each content packet with its own distinct encryption key (which Foster calls a "title key"), (3) encrypting those distinct title keys with a common key-encryption-key ("KEK"), (4) attaching the encrypted distinct title keys to their respective encrypted content packets, and (5) transmitting the encrypted content packets to receivers (*e.g.*, storage devices in the homes of consumers). Foster (EX1006), [0064], [0040]-[0041] (defining a "Recipient" as including "a receiver" and a "Receiver" as including "a consumer device, such as a set-top box, a DVD player, etc."); FIG. 8.

170. An embodiment of Foster’s method is illustrated in Figure 8 (which I have annotated below):



171. First, “content stream 100” “is parceled into content units 102.” Foster (EX1006), [0064]. Each content unit comprises a “payload” portion of the original data set (also referred to as “content packet 106”) and a “header 104,” which “includes information helpful in identifying the content” (e.g., “type of content”). Foster (EX1006), [0064]; FIG. 8.

172. Second, each payload portion is “encrypted with [a] title key 108” to “yield encrypted content packet 114.” Foster (EX1006), [0064]. Rather than using a common title key for all portions, Foster teaches that “each content packet could

be encrypted with a *different* title key” (Foster (EX1006), [0065]), which “provides enhanced security by preventing an entire content stream from being accessed with a single key” (Foster, [0004]). *See also* Foster, [0071] (“In addition, because each content packet can be encrypted with a different title key, a party who illicitly obtains a title key cannot access the entire stream. In contrast, the pirated key is only good for decrypting the content packet to which it was attached.”).

173. Third, each title key is encrypted with a common “key encrypting key 110” (a KEK, which were (and are) well-known encryption keys that I discussed above in §VII.B.7.b.iv) that is received from a storage system (database 426 discussed below). Each encrypted title key 112 is then “attached to the encrypted content as a header extension.” Foster (EX1006), [0048], [0064], [0066], FIGS. 8-10.

174. Fourth (and finally), the encrypted title key (which may be part of a header extension having additional information), encrypted content packet, and header are combined as a “processed content unit 116,” which can be “stored by a

recipient” in a receiver (*e.g.*, stored in a consumer’s set-top box). Foster (EX1006), [0040]-[0041], [0064].¹¹

175. Foster’s Figure 16 (which I have annotated below) depicts an exemplary system in which its encryption technique can be employed. Foster, [0030], [0075]. The system includes “[d]atabase 426” (annotated beige below), which “provide[s] storage for” the Key Management Block (“KMB”) ([0077]), which is “a data structure containing multiple encryptions of a key encryption key” (a KEK that I discussed above). Foster (EX1006), [0046], [0077], [0079], FIG. 16; *see also* §VIII.B.2 below (discussing database 426 further).

176. The system also includes a “source 428” (annotated green), which can be a “content owner...such as a movie studio.” Foster, [0038], [0042], [0078]. It also includes “recipient 430” (annotated red), which can be a “receiver” such as a “set-top box” or other “consumer device” that receives content from a Content Service Provider, which is “an entity, such as a cable service provider, that provides the ‘pipeline’ through which content is delivered from a content owner to a consumer.” Foster, [0039]-[0041], [0078].

¹¹ As part of transmitting encrypted content to a storage device, Foster may further divide data based on the communication protocol (*e.g.*, TCP/IP protocol), but that division of data (*e.g.*, into IP packets) is outside the scope of the asserted claims.

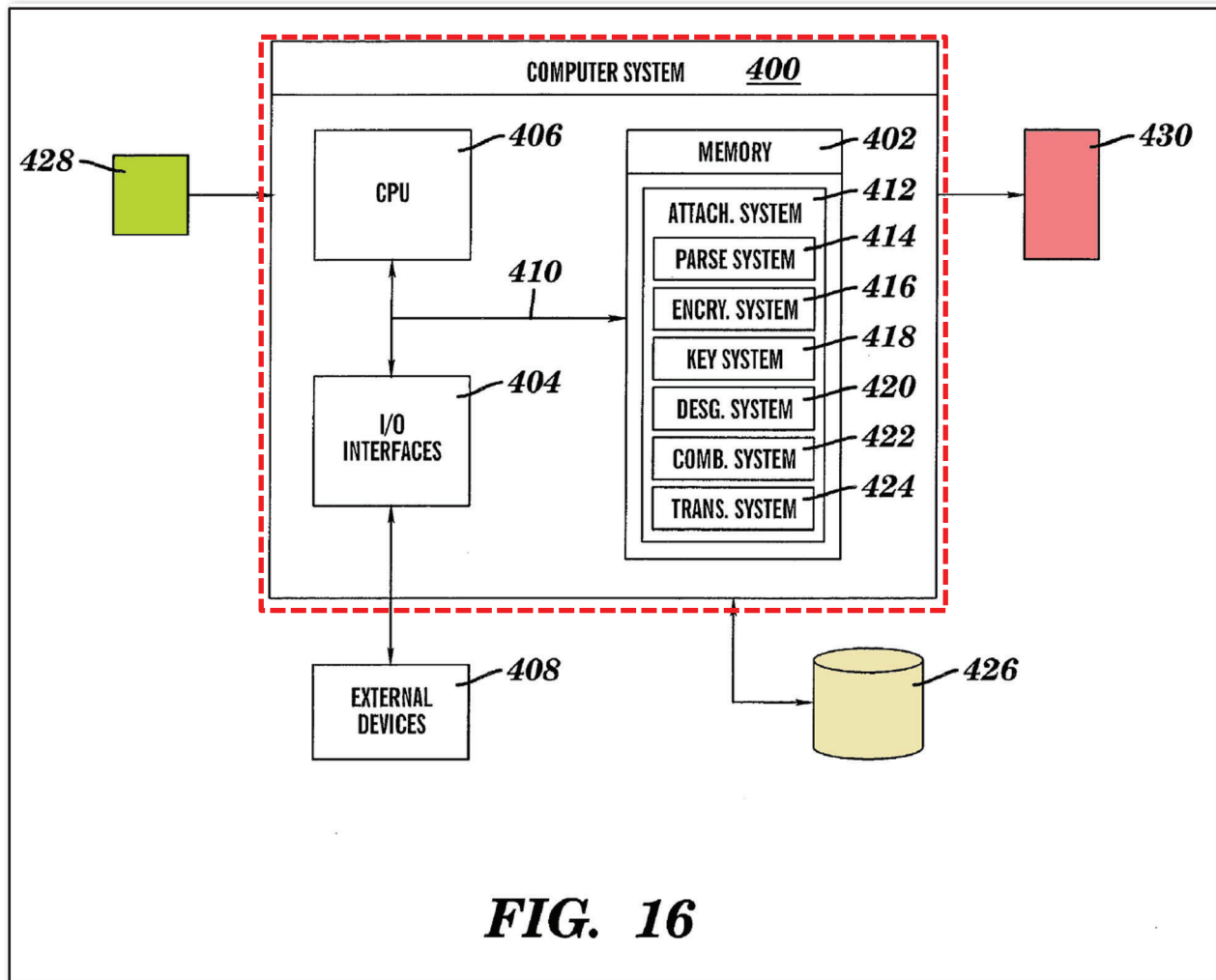


FIG. 16

177. Figure 16’s system also includes “computer system 400” (with a computer in the red-dashed box above), which is “any system capable of processing content units of an elementary content stream received from source 428.” Foster, [0078]. As system 400 is interposed between source 428 and recipient 430, POSAs would have understood Foster to disclose that computer system 400 can perform functions of the “Content Service Provider” that acts as a “pipeline” to deliver data (e.g., a digital movie) from content owner 428 to

recipients 430.¹² In other words, exemplary recipients 430 are residential customers that purchase or rent from the content service provider digital movies that are made / owned by the content owners.

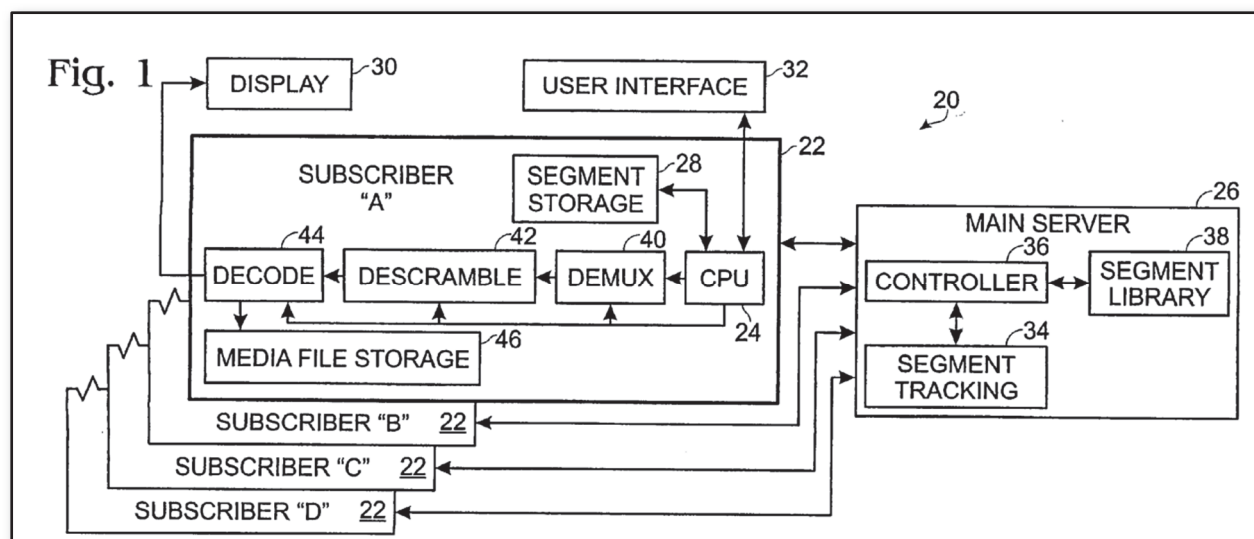
B. Hayhurst (EX1007)

178. Like Foster, Hayhurst concerns “electronic media delivery, such as video-on-demand” and how the data is “secure[ly] handl[ed] and stor[ed].” Hayhurst, [0002]. Hayhurst says then-existing systems for “delivery-on-demand” of media content “require[d] an extensive amount of bandwidth.” Hayhurst (EX1007), [0004]. To address that problem and provide other advantages, Hayhurst’s media files are segmented and “distributed among a network of subscriber units,” which store media-file segments so individual subscribers can retrieve an entire media file using segments retrieved from the various storage devices maintained at one or more different subscribers. Hayhurst (EX1007), [0006]. Like Foster’s, Hayhurst’s “segments are each encrypted ... to prevent unauthorized viewing of individual segments.” Hayhurst (EX1007), [0007].

¹² As illustrated in Figure 16, in addition to source 428 and recipient 430, there is (1) a computer (red-dashed box that I added to Figure 16 above) that performs the encryption – comprising memory 402, I/O interfaces 404, CPU 406, and bus 410 – (2) external devices 408, and (3) database 426.

179. Hayhurst’s technique advantageously (for content providers) “shifts the bandwidth requirements away from the content provider and to the network of subscriber units” ([0005]) and “provides a high level of security and a deterrence to piracy” because “[n]o single subscriber unit has more than a segment of a media file” ([0008]). Hayhurst’s technique also duplicates segments (*i.e.*, the same segment is stored on multiple subscriber units), which beneficially “provide[s] redundancy” – each segment is available even if “a given subscriber unit is unable to provide the” “segment stored on that given unit.” Hayhurst, [0009]; *see also* [0025].

180. Hayhurst’s Figure 1 (copied below) illustrates several subscribers units (subscribers A-D) “in communication with a main server 26” (*e.g.*, a server at the content service provider) and each unit is “conventional set-top box” (just like exemplary “set-top box” receivers at Foster’s recipients 430) that includes “a storage device, such as a conventional hard disk” (called a “segment storage” device) for storing “segments of one or more media files.” Hayhurst (EX1007), [0017]-[0018]; Foster, [0040] (“Receiver—a consumer device, such as a set-top box....”).



181. Main server 26 includes a “segment library 38 that is stocked 100 with segments of available movies” and “a segment-tracking module 34 for tracking the location of the segments” within the network. Hayhurst, [0023]; [0026]; FIG. 4. The server provides each subscriber unit 22 with “one or more segments of a complete media file.” Hayhurst (EX1007), [0020]. Each segment “includes data representing a small portion of the movie” and “a header that includes a packet identifier, descrambling control, and other information,” which is information that could be used to confirm that all “packets are collected.” Hayhurst (EX1007), [0020]. When a subscriber requests digital content (e.g., a movie), in one example the main server receives the request and “send[s]...instructions to the network subscriber units [having segments of the movie] to direct all of the segments of the requested movie to the requesting subscriber,” which are “processed by a decoder” at the subscriber’s device to reconstruct the digital content. Hayhurst, [0024]-

[0030]; *see also* Hayhurst, [0031] (disclosing, alternatively, using a “peer-to-peer approach whereby the request is provided directly to” the subscribers, rather than to the main server, and subscribers “responds by automatically forwarding th[e] segments to the requesting subscriber”).

C. The Foster-Hayhurst Combination

182. In my opinion, in light of Foster, Hayhurst, and the background knowledge of POSAs, POSAs had reason to store Foster’s encrypted content units on multiple storage devices (*e.g.*, consumer set-top boxes like Hayhurst’s subscriber units and Foster’s recipients 430) based on Hayhurst’s techniques for storing and distributing encrypted data segments to multiple storage devices.

183. In my opinion, POSAs would have recognized several benefits in storing Foster’s individual (encrypted) content units distributed on different storage devices as Hayhurst taught. As I discussed above (§VIII.A), Foster’s encryption method can be implemented in a computer system at a content service provider. As I also discussed above (§VIII.B), Hayhurst teaches benefits, including to content service providers, of storing media-file segments (like Foster’s content units) on multiple storage devices (like receivers at Foster’s recipients 430). For example, Hayhurst teaches that its technique advantageously reduces the content provider’s bandwidth requirements (Hayhurst (EX1007), [0005]), “provides a high level of security and a deterrence to piracy” (Hayhurst (EX1007), [0008]), and

supports redundancy in segment available from storage (Hayhurst (EX1007), [0009]). Thus, when implementing Foster's system at content service providers (*i.e.*, Foster's system sends encrypted content directly to consumers as shown, *e.g.*, in Foster's FIG. 16), Hayhurst explicitly gives POSAs reasons to apply Hayhurst's distributed-storage technique to Foster's system.

184. In addition to Hayhurst's explicit teachings about the benefits in bandwidth, security, and redundancy (also referred to as "fault tolerance"), POSAs knew of those benefits from background knowledge, and also knew that storing segmented content units on multiple storage devices in a distributed fashion would have provided additional advantages in terms of one or more of the following:

- aggregate speed – for example, by allowing segments to be retrieved from multiple sources as opposed to one storage device the speed of retrieving the stored content would have, in general, been improved. My opinion in this regard is corroborated by EX1062 at [0022]-[0023], which discloses a system similar to Hayhurst that provides a "large advantage" over storing media files in one storage device (EX1062, [0022]) because "the bandwidth needed to download content ... is vastly reduced" by distributing portions to multiple storage devices.
- efficiency / "intelligence" – for example, by being able to select storage device(s) that would provide the best delivery, based on availability,

location, etc., would have improved the reliability and speed of retrieval.

My opinion in this regard is corroborated by EX1037, which discloses at [0012] that by splitting a file “into multiple pieces across the N servers, e.g., four servers” the file “may be transferred from all of them in parallel,” thereby multiplying the potential total transfer rate. *See also* EX1037, [0003], [0013], [0020].

- security – for example, by using geographically separated storage devices it would make it more challenging for an unauthorized user to retrieve and/or intentionally corrupt all of the segments associated with the same data set.

My opinion in this regard is corroborated by My opinion in this regard is corroborated by EX1043, which discloses at [0005]-[0010] that by “storing data to be backed-up ... on different back-up servers which are spatially remote from another” it “minimis[es] the risk of unauthorised access by third parties to the backed-up data.”

- fault tolerance – for example, by storing duplicates of segments on multiple storage device the system would have been more tolerant of issues with individual storage devices (e.g., storage devices that were unavailable, suddenly left the network, had poor network conditions, etc.). My opinion in this regard is corroborated by EX1043, which discloses at [0005]-[0010] that by “storing data to be backed-up ... on different back-up servers which

are spatially remote from another” “data loss becomes very unlikely” and that additional protection against data loss occurs by distributing encrypted segmented data to multiple back-up servers in such a way that less than all segments are required to recover the original data but more than one segment is required, which gives the benefit that “[t]he breakdown of one back-up server is [] non-problematic” but because “it is not possible to recover the original data on the basis of only one data stream.” This is further corroborated by EX1050, which teaches at 1:19-22, 2:31-38, 11:11-26, and 16:6-20 that “when storage is distributed among n storage device, fault-tolerance maintains data reliability in the event of failure of some (but no more than k) of the n storage devices,” where failure includes “storage device is down, network delay is high ... the file to be stored (backed up) is corrupted or destroyed,” etc. EX1050 at 15:39-16:20 similarly teaches that “[r]edundancy can, in fact, be added by replication or encoding,” where encoding is done so that data can be restored from fewer than all data blocks. *See also* EX1037, which discloses at [0003], [0012]-[0013], [0015], [0018]-[0024] that “downloading data from multiple sources or channel in parallel” helps “provide[] reliable downloads even when the sources have poor connectivity, and is thus ideal for distributing data over unreliable networks” and provides advantages in terms of “speed,” “intelligence,” “security,” “fault tolerance,” among other

advantages. Thus, POSAs had reason to incorporate Hayhurst's distributed-storage technique with Foster's encryption system, in addition to motivation from Hayhurst's explicit teachings.

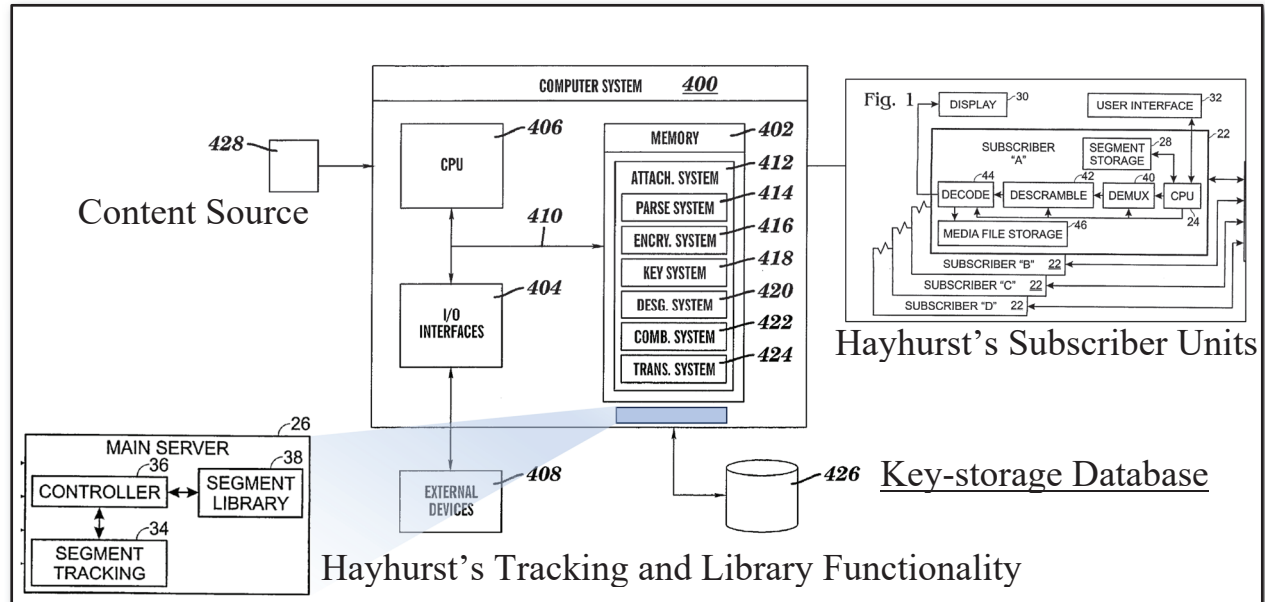
185. The Foster-Hayhurst combination that I discussed above in this section would have also been to POSAs an obvious combination of prior-art elements—combining Foster's key-storage system and encryption method to generate encrypted data with Hayhurst's method for storing encrypted data on multiple distributed storage devices—according to known encryption and data-storage methods, to yield the predictable result of either Foster's encrypted content units being stored on and retrievable from distributed storage devices as taught in Hayhurst.

186. The Foster-Hayhurst combination that I discussed above would have also been an application of Hayhurst's known distributed storage technique to Foster's known encryption system and method, which were ready for the improvement of securely storing segmented media content on multiple distributed storage devices in a manner that reduced bandwidth requirements, increased security, increased efficiency, and/or added redundancy.

187. The Foster-Hayhurst combination that I discussed above, the combination would have used Foster's technique of storing a KEK in a key-storage system and encrypting portions of a data set (*e.g.*, Foster's "payloads") with

respective encryption keys that are themselves encrypted with the KEK and Hayhurst's technique of storing the encrypted portions on different storage devices.

188. I have illustrated below one exemplary implementation of Foster-Hayhurst that is based on portions of Foster's Figure 16 and Hayhurst's Figure 1, in which Foster's "recipients 430" are implemented to incorporate functionality of Hayhurst's "subscriber units" and Foster's "computer system 400" is augmented to include the "segment library" and "segment tracking" functionality of Hayhurst's main server 26 so that computer system 400 can control distribution of and access to Foster's encrypted content portions on the "subscriber units." Hayhurst, [0017], [0023]-[0024], [0026], FIGS. 1, 4; Foster, [0040]-[0041], FIG. 16.



Foster-Hayhurst (exemplary implementation)

I note that Foster at [0055] discloses the KEK is "typically determined to a recipient concurrently with or prior to the delivery of encrypted content 26."

Conversely, Hayhurst discloses that the “encryption key to unlock the movie” stored on the main server until a subscriber’s unit sends a “request[]” to “unlock the movie,” which the “main server provides this, and billing information is then recorded in the main server.” Hayhurst, [0007]. POSAs would understand that Foster’s disclosure assumes that recipients are already authorized to decrypt the encrypted content they receive (which is why they receive the KEK before or along with the encrypted content), whereas Hayhurst’s disclosure assumes that recipients are not yet authorized to decrypt the content they receiver (which is why they have to make a request and payment to the main server to receive the encryption key). In the Foster-Hayhurst combination, the KEK would be maintained in a storage device (*e.g.*, Foster’s database 426) until requested by a subscriber, consistent with Hayhurst’s disclosure that the subscriber units are storing portions of a data set before the subscriber has been authorized to decrypt the entire data set.

189. POSAs would have reasonably expected in the Foster-Hayhurst combination that I described above because Foster and Hayhurst references both teach segmenting content for content providers and encrypting each segment. POSAs also understood that Foster’s content units are individually (randomly) accessible and agnostic to where and how they are stored compared to other content units and thus POSAs understood Foster’s content units were compatible with being stored on multiple, separate storage devices (like Hayhurst’s subscriber

storage devices). For example, Foster discloses that a “recipient” that stores encrypted content includes “a **receiver**, capable of receiving transmissions” ([0040]) and discloses that a “receiver” includes “a consumer device, such as a set-top box, a DVD player, etc.” ([0041]). *See also* Foster (EX1006), [0061] (“receipt by a recipient (e.g., content service provider or **consumer**)”). Thus, POSAs would have expected success in the Foster-Hayhurst combination because having Foster’s encrypted content units sent to and stored by multiple consumer storage devices, as taught by Hayhurst, would have been similar to Foster’s disclosure of storing its encrypted content units in a “receiver” – e.g., a “consumer device...that receives content ... **from a content service provider or from another receiver with a consumer home network.**” Foster (EX1006), [0040]; Hayhurst (EX1007), [0023]-[0029], [0031], claim 1 (method for “delivering a media file”), claim 12 (claim 1’s method for delivering further includes “encrypting each of the segments”). In other words, Foster’s content units are generally agnostic to where and how they were stored and Hayhurst’s distributed storage devices are generally agnostic to how portions of content and/or encryption keys were encrypted.

190. Thus, POSAs would have also reasonably expected success in using ordinary computer programming skill to distribute Foster’s encrypted content units to multiple storage devices based on Hayhurst’s teachings by incorporating into

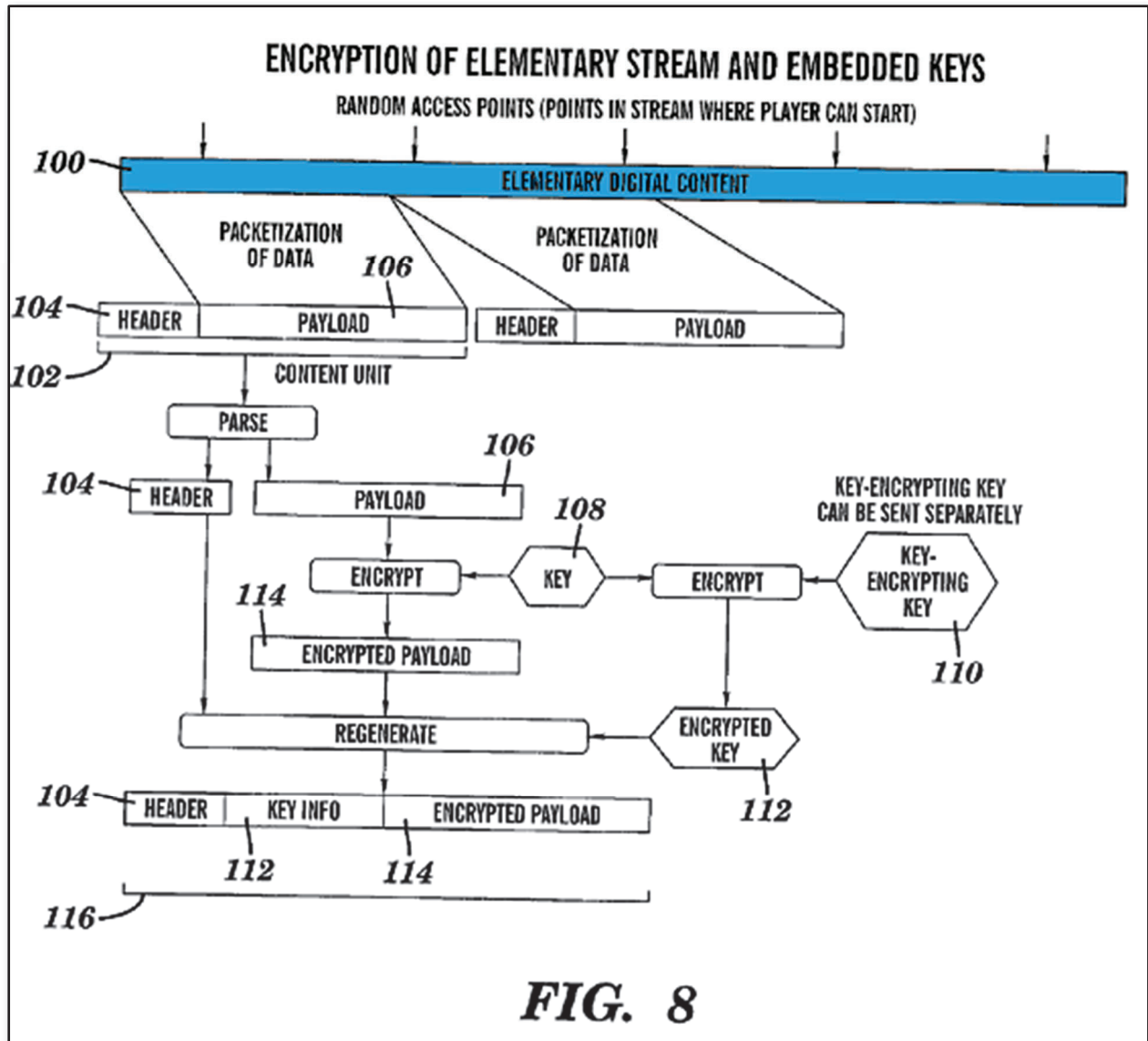
Foster’s system functionality to distribute data to multiple storage devices based on Hayhurst’s teachings.

D. Claim 1:

1. 1[PRE] A method for securely storing a data set, the method comprising:

191. The Foster-Hayhurst combination (§VIII.C above) would have performed a “*method for securely storing a data set*,” as claimed.

192. As I discussed above (§VIII.C), Foster-Hayhurst performs Foster’s method for securing digital content – for example, “video and audio data” – by encrypting it “for storage by a receiver.” Foster (EX1006), [0004], [0008], FIG. 8 (“Elementary Digital Content”), [0037] (“Content—any data such as digital image, sound or binary data deliverable from a source to a recipient”), [0050] (noting “content” is a “file”). Foster’s digital content is a “*data set*” under both parties’ constructions because POSAs understood that Foster’s digital content (*e.g.*, a digital movie, which comprises video and audio data) is a “collection of related information made up of separate elements [*e.g.*, video frames and audio samples] that can be treated as a unit [*e.g.*, a file] in data handling” (thus meeting Google’s construction; EX1064, 9-10) and “a collection of information for storage” (thus meeting Patent Owner’s construction; EX1065, 6-7).



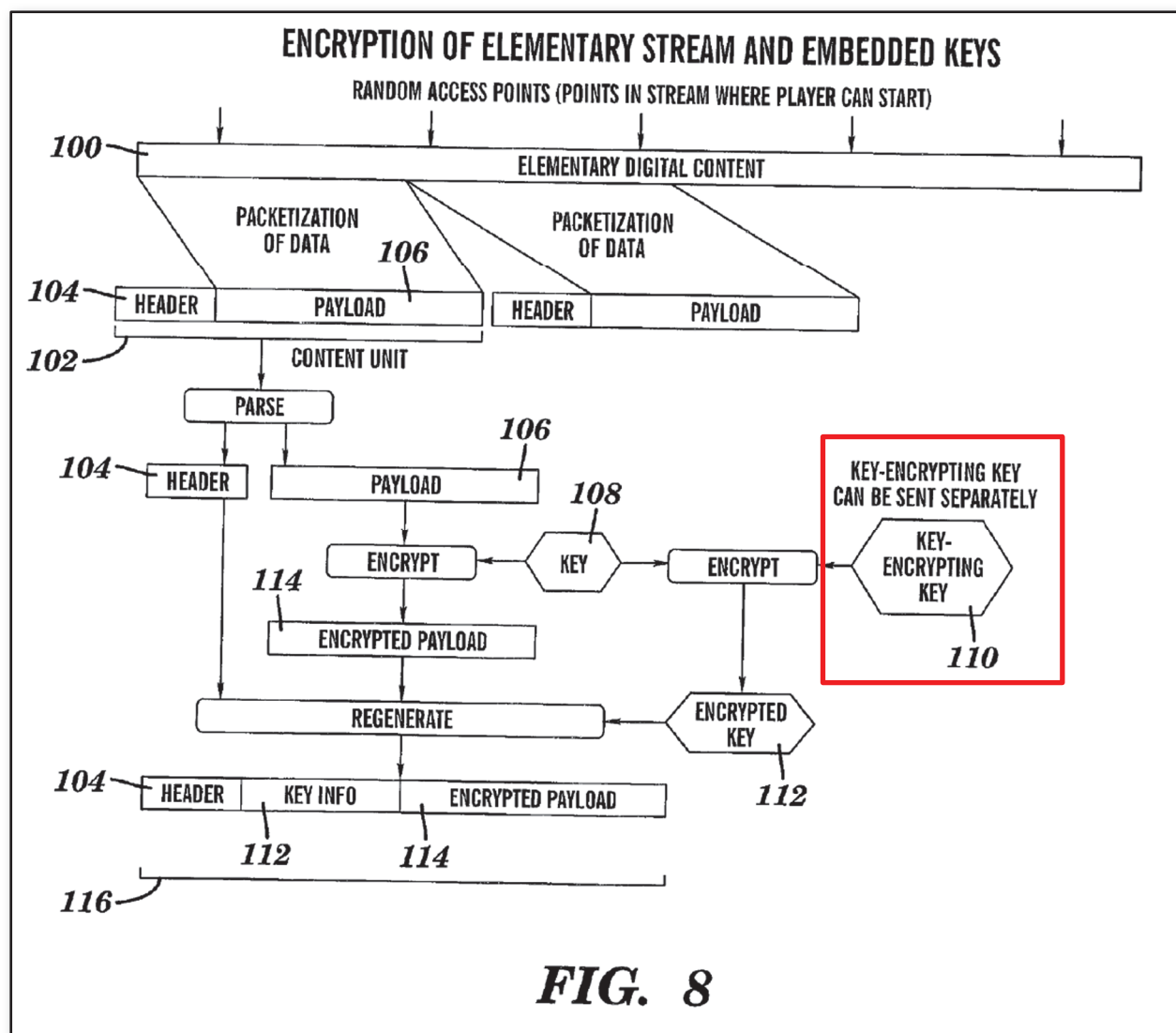
Foster, FIG. 8 (annotated)

2. [1A] receiving an external key from an external storage system,

193. As I discuss below, the Foster-Hayhurst combination (§VIII.C) would have “*receiv[ed] an external key from an external storage system,*” as claimed.

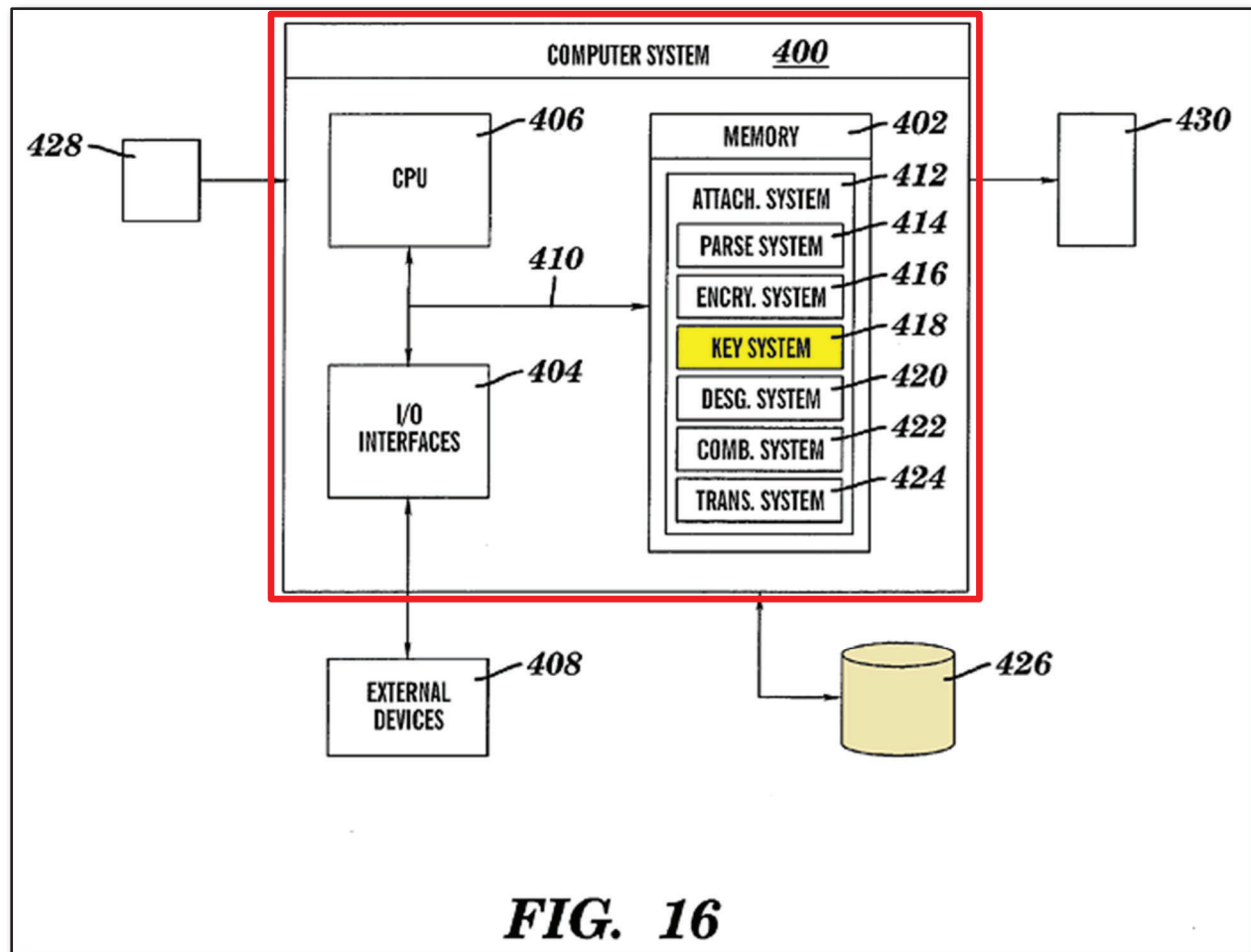
194. As I discussed above (§VIII.C), Foster-Hayhurst performs Foster’s encryption method and uses Foster’s key-storage system to store the encryption

keys received by the component performing the encryption method. As I discuss in more detail below (§VIII.D.6 [1B3] & §VIII.D.7 [1C]), Foster’s encryption method includes a step of encrypting distinct title keys (*see*) with a key encrypting key (KEK), which is (or alternatively would have been obvious to implement as) as “*external key*” for the reasons I explain below. *See also* §VII.B.7.b.iv above (discussing KEKs). This step is illustrated in Foster’s Figure 8 (which I have annotated below):



Foster, FIG. 8 (annotated)

195. As shown in Foster's Figure 16 (which I annotated below), Foster's encryption method uses a "key system 418." Foster (EX1006), [0078]-[0079], FIG. 16. Key system 418 "use[s] the key encrypting key ... to encrypt the title key[s]." Foster (EX1006), [0079] (noting that the KEK can itself be "encrypted in KMB [Key Management Block]"), [0046].



Foster, FIG. 16 (annotated)

196. “Database 426 [(annotated above)] provide[s] storage for” the Key Management Block (“KMB”) ([0077]), which is “a data structure containing multiple encryptions of a key encryption key” ([0046]). Foster (EX1006), [0046], [0077], [0079], FIG. 16. Moreover, Foster discloses that “database 426 may include ...storage devices” and be distributed across “a storage area network (SAN).” Foster (EX1006), [0077]. Because the KMB (in database 426) contains the KEK (an encryption key), database 426 is thus a (key) storage system. Foster discloses that “database 426” may be distributed across “wide area network

(WAN) or a storage area network (SAN) (not shown),” separate from computer system 400. Foster (EX1006), [0077].

197. In Foster-Hayhurst, as I discussed above (§VIII.C), content data are encrypted by computer system 400 and then stored at recipients 430 (subscriber units). As illustrated in Foster’s Figure 16 (which I color-annotated above), database 426 is a distinct component from the computer that performs the encryption – the computer in the red box I annotated above that includes memory 402, I/O interfaces 404, CPU 406, and bus 410). That computer, not database 426, encrypts the data set. Conversely, database 426 does not store or encrypt the content data; it only stores encryption keys.

198. Thus, Foster’s KEK is an “*external key*” under SFI’s proposed construction (EX1065, 8-9) because it is stored separately from the encrypted data and Foster’s database 426 (which stores the KEK) is an “*external storage system*” under the apparent scope that Patent Owner attributes to “*external storage system*” in its infringement contentions (EX1009, 2-3), as I discussed above in §VII.B.2.a.i.

199. When encrypting content data, Foster’s computer (in the red box I annotated above) receives and uses a KEK from database 426’s key-storage system, and therefore, under SFI’s constructions, Foster-Hayhurst “*receiv[es] an external key from an external storage system*” under SFI’s constructions.

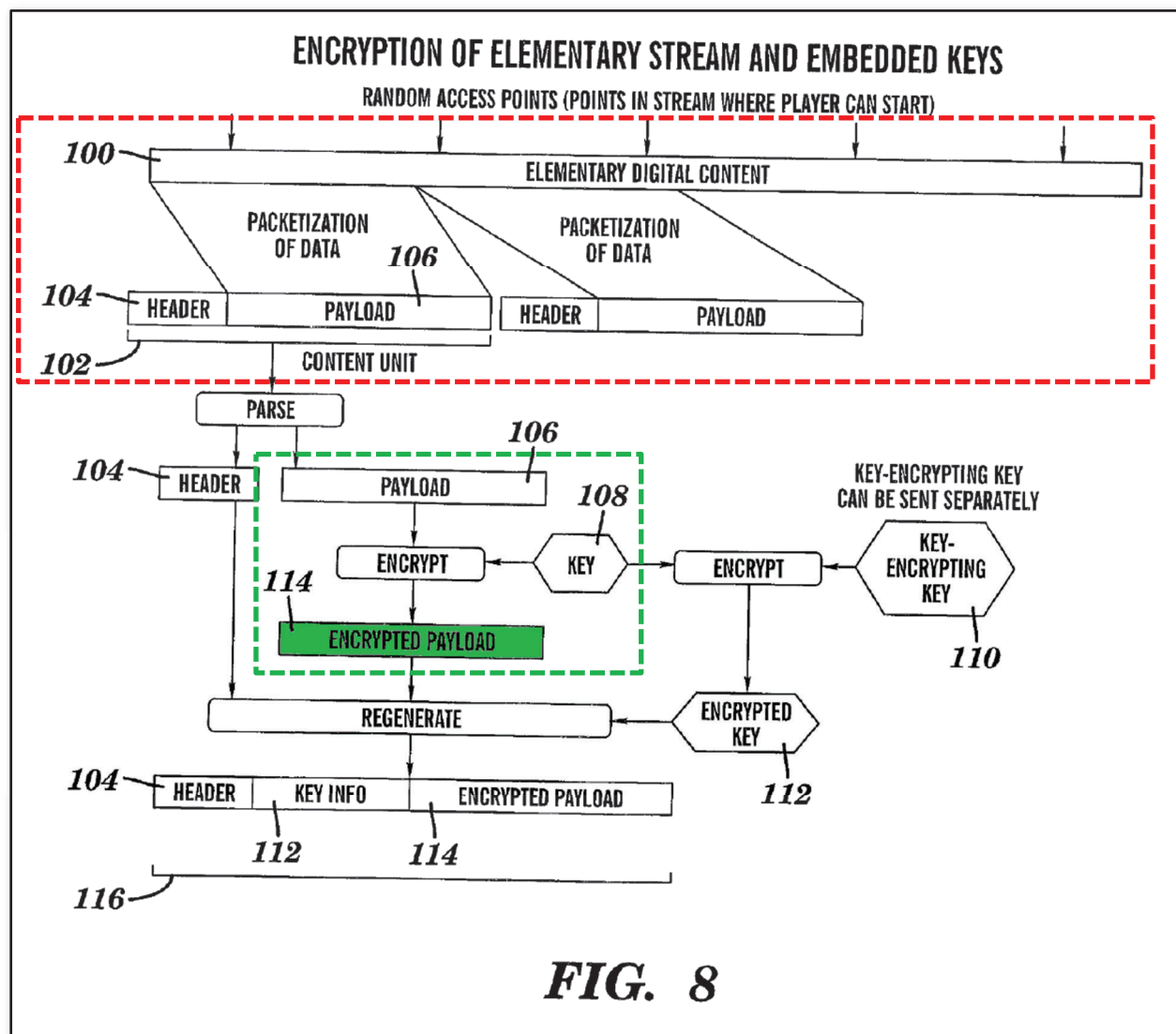
200. Under Google’s constructions of “*external key*” and “*external storage system*” (EX1064, 11-13), POSAs had reason to implement database 426 to be under the control of a third party that generates the KEK(s) and distributes it to Foster-Hayhurst’s computer because POSAs knew this implementation would beneficially reduce the burden on Foster-Hayhurst’s system by having the generation of KEKs, and distribution and authorization of KEKs to recipients (*e.g.*, after making a payment to be authorized to receive the KEK) handled by a third party’s system.

3. [1B] generating a plurality of data chunks based on the data set, such that the data set can be reconstructed using at least a minimum number of the plurality of data chunks:

201. As I discuss below, Foster-Hayhurst “*generat[es] a plurality of data chunks based on the data set, such that the data set can be reconstructed using at least a minimum number of the plurality of data chunks,*” as claimed, under Patent Owner’s proposed constructions and renders it obvious under Google’s proposed constructions.

202. As I discussed above (§VIII.C), Foster-Hayhurst performs Foster’s encryption method. As illustrated in the red box I added to Foster’s Figure 8 (annotated below), Foster generates a plurality of “payloads” by “parcel[ing]” the elementary content stream (which is the claimed “*data set,*” *see* §VIII.D.2[1A]

above). Foster (EX1006), [0064], FIG. 8; *see also* (§VIII.D.4 [1B1] below explaining how Foster’s “payloads” meet the claimed “shares”).



Foster, FIG. 8 (annotated)

203. As I discuss further below (§VIII.D.4 [1B1], §VIII.D.6 [1B3]) and is illustrated in Foster’s Figure 8 (in the green box I added above), the payloads ([1B1]’s “shares”) are encrypted to generate **encrypted payloads**, which are “a plurality of **data chunks** based on the data set,” as claimed. POSAs thus

understood Foster’s processed used in Foster-Hayhurst meets Patent Owner’s construction of the ”*generating*” step because Foster is “creating a plurality of data chunks [the encrypted payloads] based on the data set [the digital content].”

EX1065, 1.

204. As I discuss below, Foster-Hayhurst’s “*data set can be reconstructed using at least a minimum number of the plurality of data chunks,*” as claimed, under Patent Owner’s proposed constructions and renders it obvious under Google’s proposed constructions.

205. Foster further discloses that “because each **content packet** can be encrypted with a different title key, a party who illicitly obtains a title key *cannot access the entire stream.*” Foster (EX1006), [0071]. Thus, all **encrypted content packets** (the “**data chunks**”) are required to access (“*reconstruct*”) the entire stream (“*data set*”). See Foster, FIG. 8. Therefore, the elementary content stream (the claimed “*data set,*” §VIII.D.2 [1A]) “*can be reconstructed using at least a minimum number of the plurality of data chunks,*” under Patent Owner’s construction of “*minimum number of...chunks,*” which includes reconstructing the data set using “the total number of data chunks.” EX1065, 13; *see also* EX1009, 4-5 (arguing limitation is met by reconstructing data from “all the chunks in which that data is stored”).

206. Under Google’s construction, POSAs had reason to distribute Foster-Hayhurst’s “*data set*” into encrypted payloads (“*data chunks*”) such that the data set could be reconstructed “*using at least a minimum number of the plurality of data chunks*” that is “a predefined number of the lowest amount of the plurality of data chunks.” EX1064, 13. For example, POSAs would have had reason to distribute the data set into data chunks with enough redundancy such that some predetermined number of data chunks (*e.g.*, 10 of 100 data chunks) could be lost (or inaccessible or corrupted) and yet the original data set could still be recovered (*e.g.*, by having 90 out of 100 data chunks). This would have been a conventional way to distribute and store data by the early 2000s and it would have provided known benefits – like redundancy and protection against loss of / inability to retrieve individual data chunks. My opinion in this regard is corroborated by Orsini (at [0280]-[0284]), which discloses limitation [1B] for the reasons I discussed above (§VII.B.3 (citing EX1050 and EX1052)).

207. My opinion in this regard is corroborated by EX1050, which discloses at 2:31-38 as background information a technique that “enhance[s] the reliability and availability” of data in a “network-based distributed storage system” that “add[s] redundancy to the original data” such that data is distributed into blocks such that “the actual data can be restored” even if there are some server failures. EX1050 at 11:11-26 discloses a similar technique “to protect the system from loss

of data [that] allow[s] reconstruction of the original data even in the absence of some data blocks” (e.g., “The distribution is such that if a number of backup partners are lost, and that number is smaller than or equal to $K(\leq k)$, it is still possible to recover any block by using data blocks and redundant data blocks from remaining backup partners.”). *See also* EX1050, 1:19-22, 15:39-16:20. My opinion is further corroborated by EX1052 at [0001], which discloses as background information that “[m]aintenance of multiple copies of data is part of the security function in data processing operations in case data is unavailable, damaged, or lost.”

208. Distributing Foster’s digital content (“*data set*”) into encrypted payloads (“*data chunks*”) such that the data set could be recovered from fewer than all chunks would have also been applying a known technique (of data distribution and storage) to a known device (Foster’s encryption device) ready for the improvement of redundancy and the ability to recover data when one (or more) chunks are lost, corrupted, and/or unretrievable.

209. POSAs would have had a reasonable expectation of success because those data-distribution techniques were well-known and conventional by the mid-2000s and based on precise mathematics. My opinion in this regard is corroborated by EX1050 (discussed above) at 1:19-22, 2:31-38, 11:11-26, 16:6-20 and in §VIII.D.8 [1D] below, where I cite corroborating evidence of the benefits of

distributing encrypted portions of data to a plurality of storage devices such that the original data can be recovered from more than one but fewer than all encrypted portions.

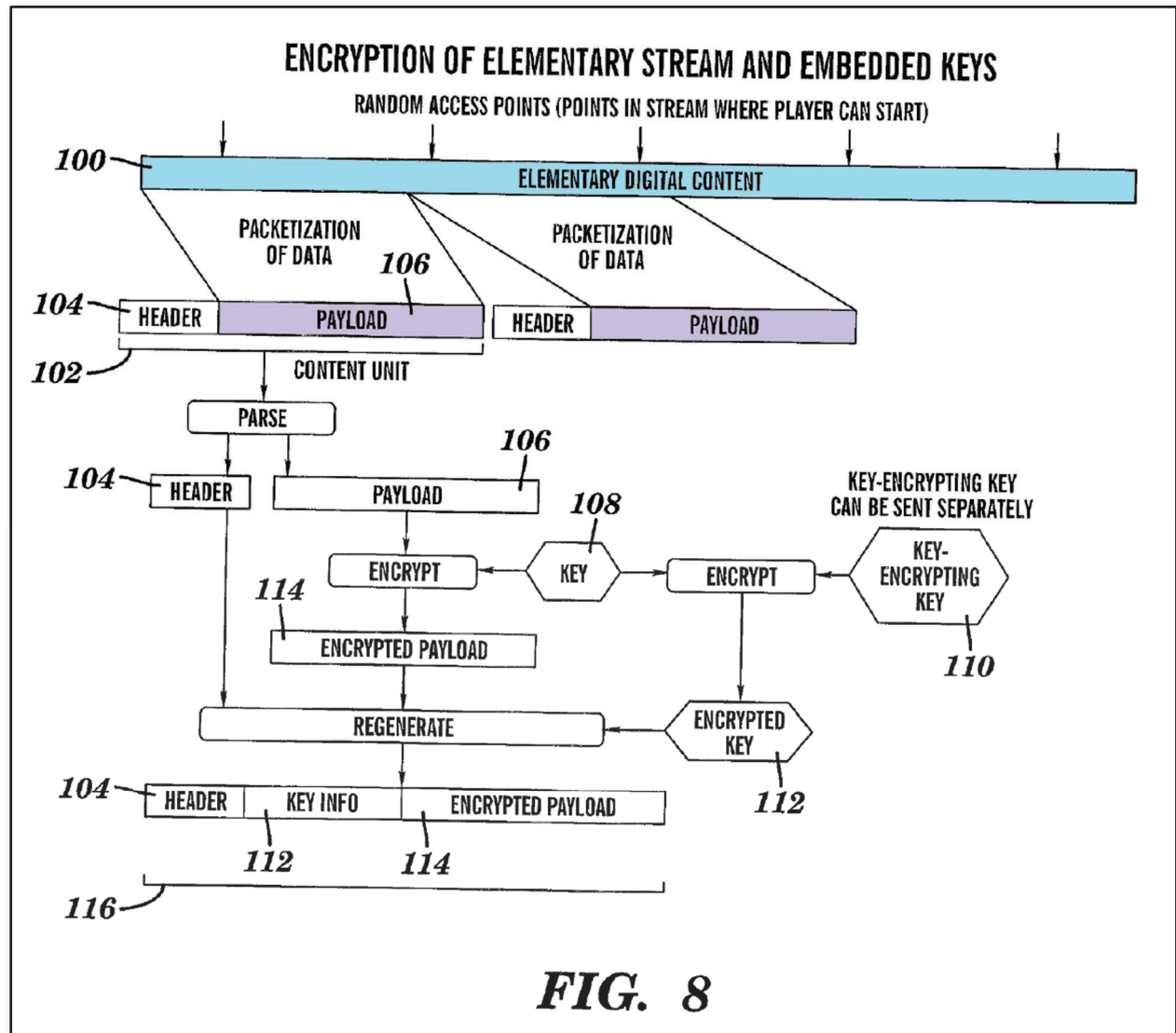
**4. [1B1] wherein generating the data chunks comprises:
distributing the data set into a plurality of shares, wherein
each of the shares comprises less than all of the data set,**

210. As I discuss below, Foster-Hayhurst's process for "*generating the data chunks*," comprises "*distributing the data set into a plurality of shares, wherein each of the shares comprises less than all of the data set*," as claimed.

211. As I discussed above (§VIII.C), Foster-Hayhurst performs Foster's encryption method. As illustrated in Foster's Figure 8 (which I have annotated below), Foster "parcel[s]" ("*distribut[es]*") the **elementary content stream** ("*data set*") into a plurality of "**payloads**" ("*shares*"). Foster (EX1006), [0064]. Foster thus meets both parties' proposed constructions of "*distributing the data set...*" because Foster is "splitting" (Google's proposed construction, EX1064, 2-5) and "dividing and placing" (Patent Owner's proposed construction, EX1065, 2) "the data set into a plurality of shares."

212. As Foster's Figure 8 illustrates, each **payload** "*comprises less than all of the data set*," as claimed. This is because, as Foster discloses, the entire stream (data set) cannot be recovered from the payload of a single content packet (Foster

(EX1006), [0071]) and so the data in each payload/**share** “comprises less than all of the data set,” as claimed.

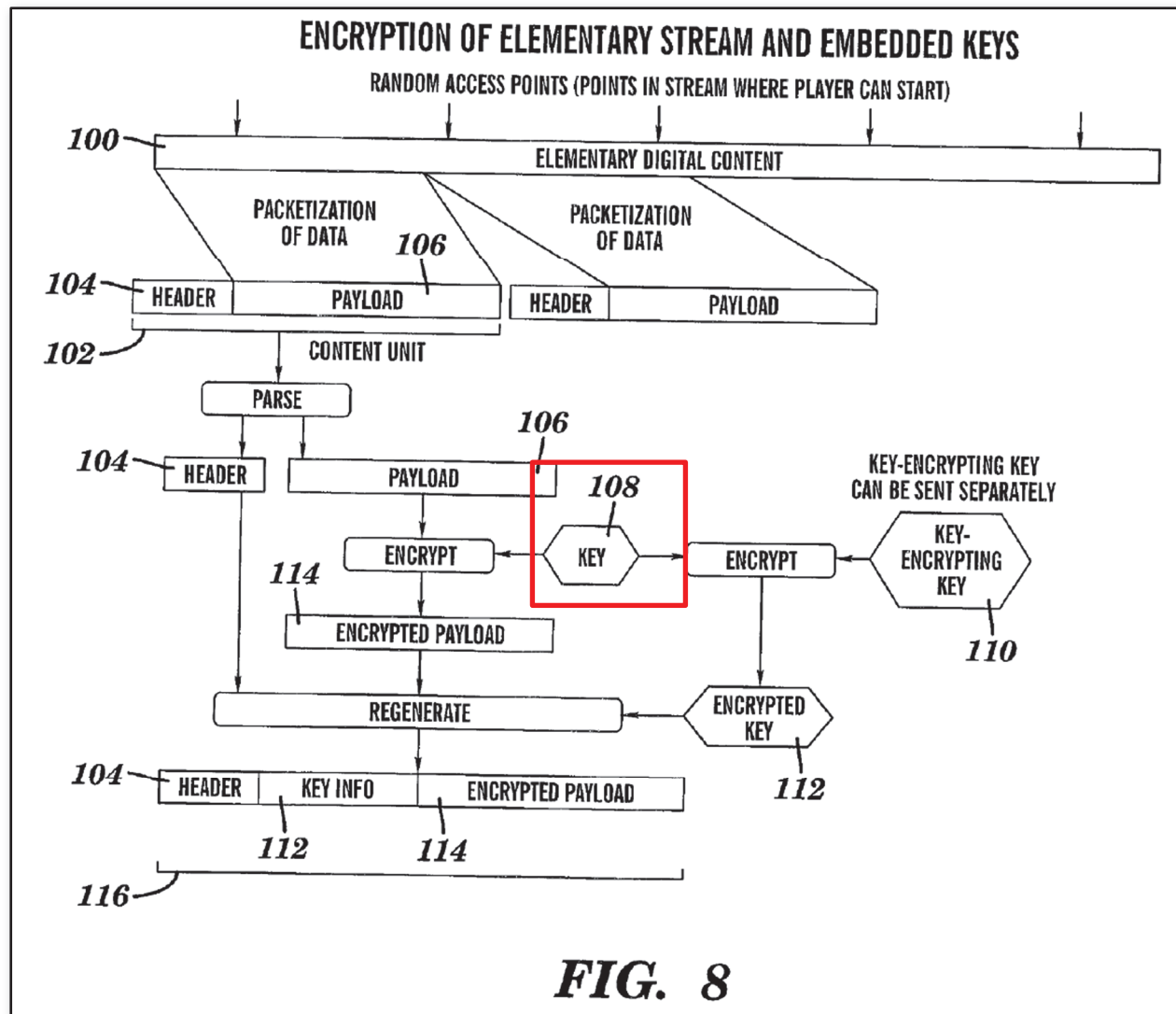


Foster, FIG. 8 (annotated)

**5. [1B2] [wherein generating the data chunks comprises:]
accessing a plurality of distinct encryption keys,**

213. As I discuss below, Foster-Hayhurst's process for "*generating the data chunks*," further comprises "*accessing a plurality of distinct encryption keys*" as claimed.

214. As I discussed above (§VIII.C), Foster-Hayhurst performs Foster's encryption method. As illustrated in Foster's Figure 8 (which I annotated below), Foster-Hayhurst's encrypting device uses title keys (the "*a plurality of distinct encryption keys*") to encrypt each payload (each of which is a "*share*," see §VIII.D.4 [1B1] above) according to Foster's encryption method. Foster (EX1006), [0064]-[0065] ("each content packet could be encrypted with a different title key"). Thus, Foster-Hayhurst would have opened a file containing the title keys and/or retrieved the title keys from storage, thereby "*accessing*" them (*e.g.*, accessing them from Foster's key database 426 or a conventional key-generating/key-storage device).



Foster, FIG. 8 (annotated)

6. [1B3] [wherein generating the data chunks comprises:]
 encrypting each of the shares with a respective one of the
 plurality of distinct encryption keys,

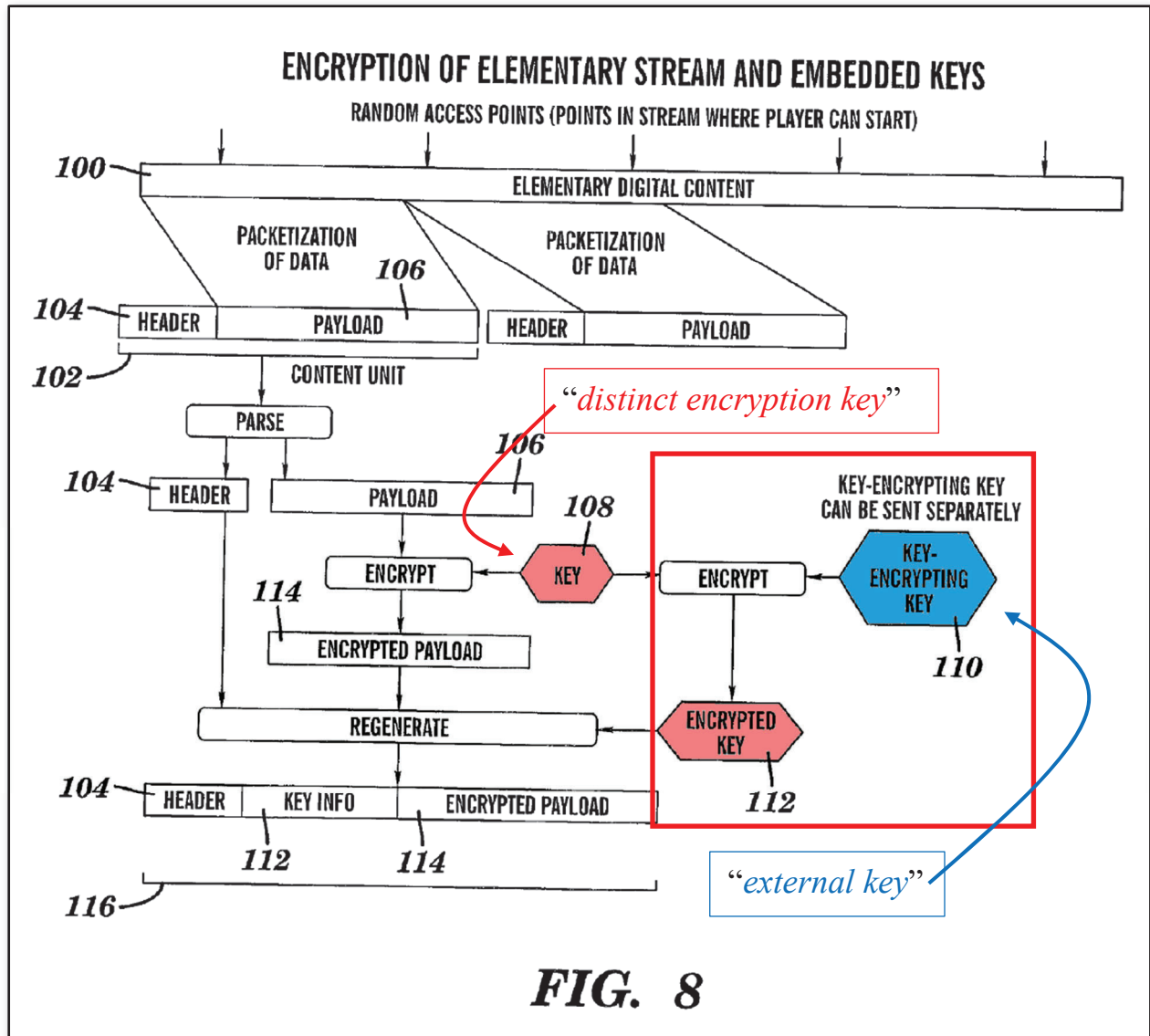
215. As I discuss below, Foster-Hayhurst's process for "*generating the data chunks*" further comprises "*encrypting each of the shares with a respective one of the plurality of distinct encryption keys,*" as claimed.

216. As I discussed above (§VIII.C), Foster-Hayhurst performs Foster’s encryption method. Using Foster’s method, “each content unit could be encrypted with a *different* title key.” As illustrated in Figure 8 above, each content unit comprises a payload, which is a “*share*” (see §VIII.D.4 above). Thus, “*each of the shares*” is encrypted with “*one of the plurality of distinct encryption keys*,” as claimed. Foster (EX1006), [0065], [0071] (a “pirated [title] key is only good for decrypting the content packet to which is was attached”).

7. **[1C] performing an encryption operation based on the external key to further secure the plurality of data chunks; and**

217. As I discuss below, Foster-Hayhurst “*perform[s] an encryption operation based on the external key to further secure the plurality of data chunks*,” as claimed.

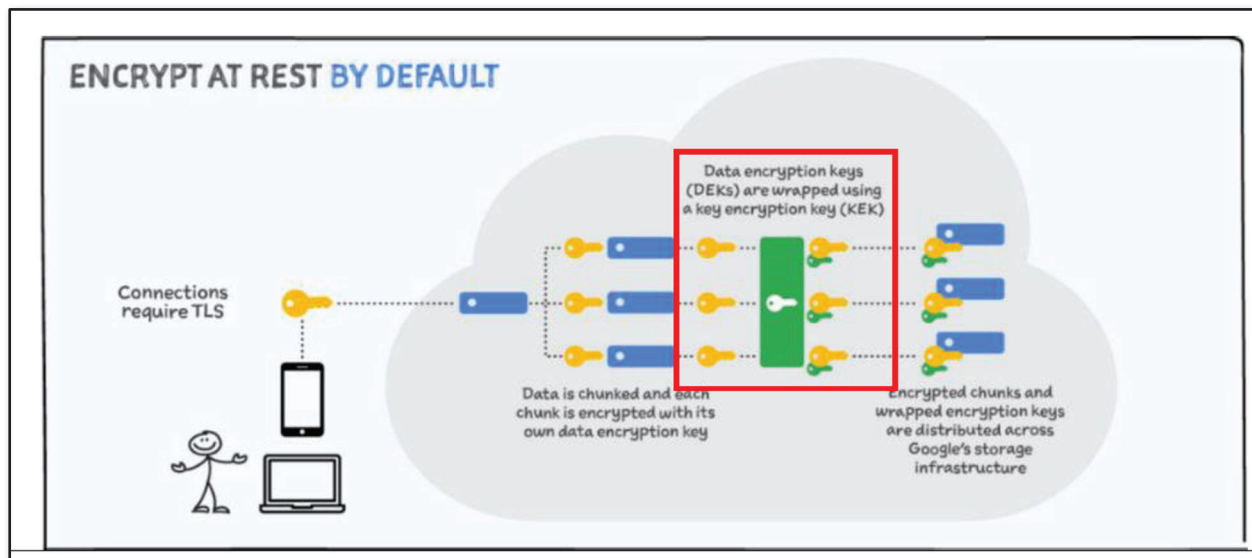
218. As I discussed above (§VIII.C), Foster-Hayhurst performs Foster’s encryption method and receives Foster’s key-encryption-key from an external storage system (e.g., Foster’s database 426 or a similar external storage device, see §VIII.D.2 [1A] above). Foster’s Figure 8 (which I have annotated below) illustrates encrypting each **title key** with the **KEK** (which is the claimed “*external key*,” see §VIII.D.3 [1B]), which is an encryption operation based on the KEK. Foster (EX1006), [0064]-[0065].



Foster, FIG. 8 (annotated)

219. As shown below, under the scope of the claims that Patent Owner attributes to [1C] in its district court complaint, encrypting [1B3]’s title keys with a KEK meets [1C]. See EX1009, 8 (arguing “DEKs [data encryption keys] are encrypted with (wrapped by) a key encryption key (KEK)”); §VII.B.7.b.v (Orsini’s

[1C]). Thus, Foster meets [1C] (at least) under Patent Owner’s assertion of the claim scope.



EX1009 (Patent Owner’s Infringement Contentions), 8

220. Alternatively, [1E] is met because POSAs also had motivation to reorder data (*e.g.*, bits or bytes of data) that is within each “*data chunk*” (the encrypted payloads) using an “*external key*” (*e.g.*, the KEK or a random number generated from the KEK) to further secure the plurality of data chunks, for the benefit of additional security that I discussed above in §VI.B.7.b.iii (EX1031, EX1032, EX1040, EX1041). As I discussed above in §VI.B.7.b.iii with cited corroboration, this would have been a conventional encryption technique to further secure data. As I discuss in §VII.G below, POSAs had reason to reorder the data within each of Foster’s shares *before* encrypting each share based on the “*external key*.” POSAs would have understood it would have been as obvious to reorder the

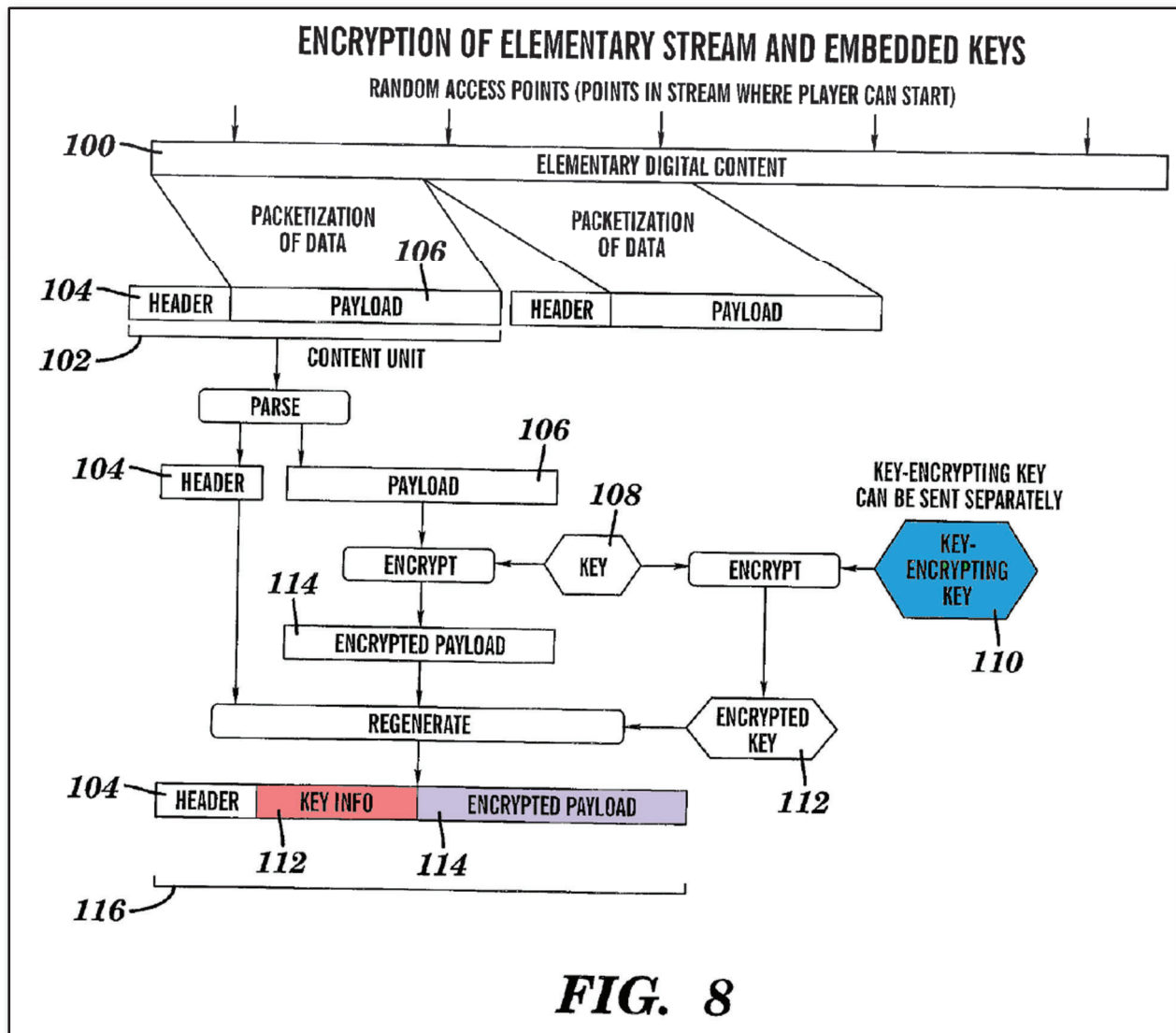
data within each share *after* encrypting each share because POSAs understood that “employing a stream cipher after separation of the fragments into separate fragments is *logically equivalent to employing the stream cipher on the initial fragment stream* before its separation into separate fragment streams. In each case, the fragments are individually enciphered by the stream cipher process.” EX1028 (filed 2001), [0090]. See also the prior art I discussed above in §VI.B.7.a.i (EX1054, [0026]; EX1045, [0109]-[0113, FIGs. 11-13) which further corroborate my opinion that reversing the order of when an encryption step is performed in an encryption method is generally a design choice and was conventional. Because reordering the bits/bytes of data (encrypted or otherwise) was conventional, well known, and an exact mathematical process POSAs would have reasonably expected success in reordering the data within each of Foster’s encrypted payloads based on the KEK.

8. [1D] storing with the plurality of data chunks data indicative of at least one of the distinct encryption keys on a plurality of different storage devices.

221. As I discuss below, Foster-Hayhurst “*store[s] with the plurality of data chunks data indicative of at least one of the distinct encryption keys on a plurality of different storage devices,*” as claimed.

222. As I discussed above (§VIII.C), Foster-Hayhurst performs Foster’s encryption method. As illustrated in Foster’s Figure 8 (which I have annotated

below), Foster’s method generates and stores a “processed content unit 116,” which includes three pieces of information: (1) an encrypted **payload** 114 (the “*data chunk*”); (2) “encrypted **title key** 112” (the “*distinct encryption key*”), which is a title key that “has been encrypted with the **KEK**” (the “*external key*”); and (3) header 104. Foster (EX1006), [0065]-[0066], [0071] (“[T]itle keys 108 and 208 are attached and transmitted with the content packets they were used to encrypt.”), FIG. 9 (which “depicts a header extension,” ([0023])). In particular, the **title keys** (which meet the claimed “*distinct encryption keys*,” *see* §VIII.D.6 above) will be encrypted with the KEK (which meets the claimed “external key,” *see* §VIII.D.7 above). In the Foster-Hayhurst combination, each “processed content unit 116” is stored on a receiver at a different “receipt 430.” POSAs understood each receiver (*e.g.*, consumer set-top box) includes a “*storage device*,” also the combination’s recipients are implemented with the functionality of Hayhurst’s subscriber units, including their storage devices. §VIII.C.



Foster, FIG. 8 (with coloring added)

223. Thus, the plurality of “processed content units” (the “*data chunks*”) are stored “*on a plurality of storage devices,*” as claimed.

224. Under Patent Owner’s claim constructions and the scope Patent Owner attributes to [1D] in related district court litigation, [1D]’s “*data indicative of at least of the distinct encryption keys*” is met by one of [1B3]’s keys itself. EX1065, 3-4; EX1009, 9 (alleging limitation is met “by virtue of storing DEKs

[data encrypting keys alleged to meet [1B3]] with the data chunks”). Because Foster-Hayhurst stores each “processed content unit 116” – including a distinct title key (which meets a claimed “*distinct encryption key*”) with its corresponding encrypted payload (which meets a claimed “*data chunk*”) – Foster-Hayhurst meets [1D] under Patent Owner’s claim constructions.

225. Like Orsini, POSAs understood Foster was also compatible with various encryption algorithms, some of which had multiple encryption modes and/or supported variable-length encryption keys. See §VII.B.8 (citing EX1039). Thus, POSAs also had reason to store each content unit with “*data indicative of*” of a title key under Google’s construction by storing each of Foster’s payloads with metadata indicating the encryption mode and/or title key’s length, for the same reasons I discussed above in §VII.B.8 that POSAs had reason to store Orsini’s encrypted shares with metadata about Orsini’s share encryption keys.

226. As I discussed above (§§VII.B.3-VII.B.8), Foster performs limitations [1B]-[1D] in the order in which claim 1 recites those steps. As to [1A], Foster does not limit when it receives a KEK. It would have been a conventional design choice to receive the KEK as a first step in Foster’s method (*e.g.*, when beginning the steps to encrypts a data set of digital content). Additionally, when encrypting several data sets back-to-back with the same KEK, POSAs would have understood

that the system has already received the KEK from encrypting a previous data set, before it encrypts the next data set. Either alternative meets Google's construction that claim 1's steps be performed in order. EX1064, 16.

E. Claim 2: The method of claim 1, wherein storing with the plurality of data chunks data indicative of at least one of the distinct encryption keys comprises storing each of the plurality of data chunks with at least one of the distinct encryption keys.

227. Foster-Hayhurst meets claim 2 under Patent Owner's assertion of the claim scope for the reasons it meets [1D] under Patent Owner's assertion, as I discussed above (§VIII.D.8). Specifically, Foster-Hayhurst stores with each "*data chunk*" (e.g., Foster's encrypted payload) the title key ([1B3]'s "*distinct encryption keys*") used to encrypt that data chunk, which is "*data indicative*" of a distinct encryption key under at least Patent Owner's claim scope, as I discussed above (§VIII.D.8 [1D])). Foster (EX1006), [0065]-[0066], FIGS. 8-9; EX1009, 10-11 (alleging claim 2 is met by "storing a DEK with each data chunk"). Thus, Foster-Hayhurst meets Claim 2 under the scope of the claims that Patent Owner attributes in the related district court litigation.

- F. Claim 3: The method of claim 1, wherein storing each of the plurality of data chunks with at least one of the distinct encryption keys comprises including with a particular data chunk of the plurality of data chunks data indicative of an encryption key that was used to encrypt a different data chunk of the plurality of data chunks.**

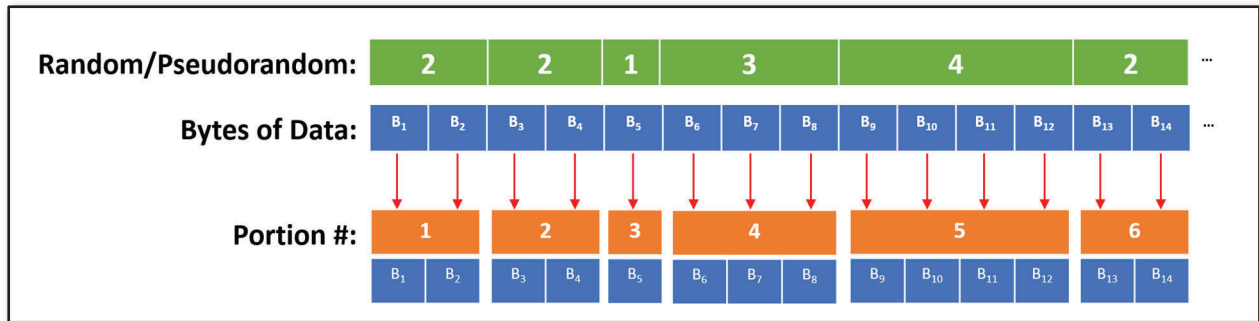
228. As I discussed above (§VIII.D.8 [1D]), each encrypted payload (“*data chunk*”) in Foster-Hayhurst is stored with the title key (“*distinct encryption key*”) used to encrypt that payload. However, POSAs had reason to modify Foster to store each encrypted payload with a title key “*used to encrypt a different data chunk*,” which is “*data indicative of an encryption key*,” under Patent Owner’s assertion of the claim scope in related district court litigation (as I discussed above, §VIII.D.8). POSAs knew storing each encrypted payload with the title key used to encrypt another payload would have had the benefit of increased security (*e.g.*, unauthorized users could not decrypt the data without knowing which payload is stored with which key). To the extent this would have increased the time to recover the original data set, POSAs understood that balancing increased security versus decreased convenience and/or increased processing time was a well-known design-choice tradeoff in encryption-method design. My opinion that POSAs understood it was conventional to tradeoff convenience for security is corroborated by EX1061, which states at [0059] that “there is a tradeoff between the security and the efficiency of computation.”

229. Storing each encrypted payload with the title key used to encrypt another payload would have also been a simple substitution of one known encryption technique (storing a payload with a different payload's encryption key) for another (storing a payload with its own encryption key) to obtain the predictable result of storing each encrypted payload (the “*data chunk*”) with another payload's encryption key. My opinion in this regard is corroborated by EX1045, FIGS. 12-13, [0111]-[0116].

G. Claim 4: The method of claim 1, wherein distributing the data set into a plurality of shares comprises using a random technique or pseudorandom technique.

230. For the reasons I discuss below, distributing Foster's data set into payloads (the “*shares*”) “*using a random or pseudorandom technique*,” as claimed, would have been an obvious implementation of Foster-Hayhurst. In particular, below I identify three well-known, conventional techniques for distributing a data set into payloads “using a random or pseudorandom technique” that would have been obvious in Foster-Hayhurst to provide additional security (*e.g.*, by making it harder for attackers to analyze/predict the structure of encrypted chunks).

231. First, POSAs had reason to divide the data set into random or pseudorandomly sized payload portions (as illustrated below) as a well-known conventional technique for providing additional security by making it more challenging for unauthorized user to recover the data set:



Foster (EX1006) at [0072] discloses embodiment where packet length is not fixed, which suggests varying the length of packets and corroborates my opinion that POSAs would have reasonably expected success in dividing Foster-Hayhurst data set into random or pseudorandomly sized payload portions. Distributing the data set in this manner would still allow for Foster’s self-identified benefit of synchronization between encrypted portions of content and respective encryption keys. To the extent this distribution added complexity to recovering the data set, such tradeoffs in security versus convenience were well-known and conventional tradeoffs in encryption methods, which is corroborated by EX1061, which states at [0059] that “there is a tradeoff between the security and the efficiency of computation.”

232. My opinion that POSAs had reason and were aware of the benefits of techniques like those discussed above is corroborated by at least the following:

- EX1044 at [0016]-[0018], which discloses “divid[ing]” data “into a plurality of segments and each segment is stored in a new location” and “to further improve the level of obfuscation, the segments are of random or pseudo-

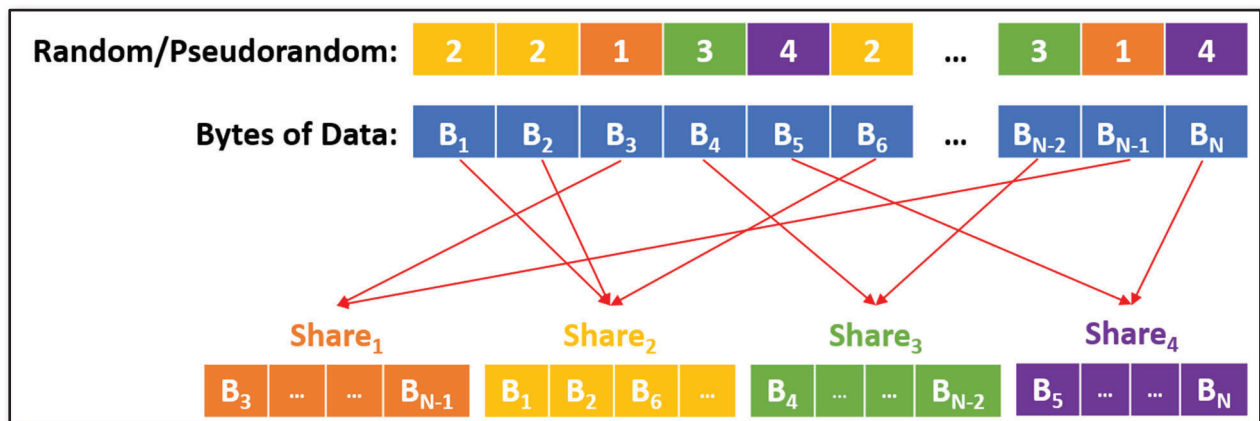
random size”). EX1044 at [0077] further discloses that “an additional level of security is provided ... by segmenting the files in random chunks and distributing these across directories[] makes it further difficult for unauthorized access to the content of these files”;

- EX1045 at FIGS. 9-10, [0103]-[0108], which discloses “divid[ing]” “data portions of different arbitrary lengths”;
- EX1046 at [0003], which discloses as background that encrypting data with “fixed size block[s] of data” can cause “the resulting encrypted data [to] include repetitive patterns that make the encrypted data susceptible to various crypto-analysis techniques which can readily decipher the encrypted data”). And EX1046 at [0021] discloses a technique for “any type of content including: text data, image data, sound data, or video data” where the data “can comprise unencrypted data or data previously encrypted by another encryption method” and the technique include “divid[ing the data] into a plurality of blocks of data” and each block “comprise[s] a random number of bits of data”; and
- EX1028 at [0015]-[0017], which discloses that the “size of fragments can be varied” thereby preventing “frequency analysis” and “known plaintext

attacks are made greatly more difficult while brute-force attacks are greatly increased in complexity.”

233. Second, POSAs also had reason to randomly or pseudorandomly distribute data into payloads (the “*shares*”) by rearranging the data within each payload (the “*share*”), also a well-known encryption technique used for additional security benefits, as I discussed above in §VII.B.7.b.iv (with corroboration cited in EX1028, EX1044-EX1046).

234. Third, it was well-known to POSAs that distributing bits/bytes of the data set into payloads (the “*shares*”) according to a random or pseudorandom number (*i.e.*, using a random or pseudorandom encryption key, as illustrated below) provided an additional layer of cryptographic security when distributing the data set into payloads (the “*shares*”).



My opinion in this regard is corroborated by at least the following:

- EX1047 at [0056]-[0058], which discloses creating “fragments from multiple files [that] are randomly mixed together and further disguised to form composite files,” which are “then sent to a plurality of hidden locations, such as the Internet, local area networks,” etc. and teaching that this randomization “can find broad application and contribute to enhanced handling of privacy-related issues on the Internet”;
- EX1048 at [0008], which discloses “randomly divid[ing] the data structure into a plurality of data fragments and [] stor[ing] the plurality of data fragments”;
- EX1045 at FIGS. 9-10, [0103]-[0108], which discloses dividing data “into a plurality of data portions” “according to a function of, for example, random numbers”; and
- EX1049 at 10:8-15, which discloses dividing an original data set S into partial data sets $S(j) = S(1), S(2), \dots S(n-1)$ “by carrying out the exclusive OR (XOR) calculation of the original partial data $[S(j)s]$ and the random number partial data $[R(j)s]$.”

Thus, Foster-Hayhurst meets Claim 4.

H. Claim 5: The method of claim 1, wherein distributing the data set into a plurality of shares comprises using a deterministic technique or pseudorandom technique.

235. Foster-Hayhurst meets claim 5 for the same reason Foster-Hayhurst meets claim 4's limitation of "*distributing the data set it into a plurality of shares comprises using a random or **pseudorandom technique**,*" as I discussed above (§VIII.G, claim 4). The '854 patent discloses that an example of a "deterministic algorithm" for distributing the data set includes "breaking up the original data into sequential shares." '854 patent (EX1001), 75:11-19. Foster-Hayhurst would have distributed the data into shares using a similar "*deterministic technique*" based on Foster's disclosure of segmenting the data set sequentially, which is illustrated in Foster's Figure 8. *See also* Foster (EX1006), [0065].

I. Claim 6: The method of claim 1, wherein each share of the plurality of shares comprises a substantially random distribution of a subset of the data set.

236. As I discussed above with corroboration (§VIII.G, Claim 4), POSAs had reason to apply to Foster-Hayhurst well-known techniques using a random or pseudorandom (*i.e.*, substantially random) number/key to cryptographically determine (i) the size of each payload (a "*share*"), (ii) what payload a portion of the data set would be distributed to, and/or (iii) the arrangement of data within each payload, any of which would have resulted in "*each share of the plurality of*

shares compris[ing] a substantially random distribution of a subset of the data set,” as claimed.

J. Claim 7: The method of claim 1, wherein performing an encryption operation based on the external key to further secure the plurality of data chunks comprises encrypting primary data with the external key.

237. As I discussed above (§VIII.C), Foster-Hayhurst performs Foster’s encryption method. As I also discussed above (§VIII.D.7, [1C]), Foster discloses encrypting the title key for each encrypted payload (the claimed “*data chunk*”) with a KEK (the claimed “*external key*”). And Patent Owner has asserted that encrypting a data chunk’s encryption key with a KEK is “further secur[ing]” the data chunk. EX1009, 8-9. If “*primary data*” is the data that comprises the original “*data set*” (e.g., Foster’s “elementary digital content”) then Foster-Hayhurst meets claim 7 under Patent Owner’s assertion of the claim language’s scope by encrypting the titles with the KEK.

238. If Claim 7 more narrowly requires performing an encryption operation directly on “*primary data*” using the external key (and “*primary data*” is the original “*data set*”), doing so by encrypting Foster’s data set using Foster’s KEK or an alternative (conventional) encryption key retrieved from an external storage system (e.g., Foster’s database 426) would have been an obvious implementation of Foster-Hayhurst. POSAs would have had motivation and a reasonable expectation of success in doing so because it was conventional and would have

provided additional security for each of Foster’s encrypted payloads (the “*data chunks*”) beyond encrypting its title key with Foster’s KEK.

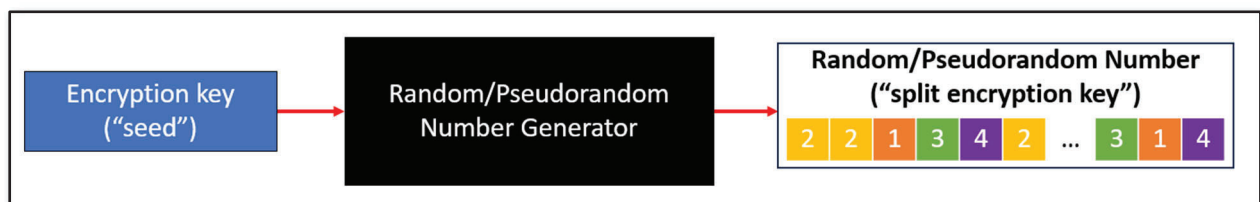
239. As I also discussed above with corroboration (§VIII.G, claim 4), POSAs had reason to use an encryption operation to distribute data into payloads (the “*shares*”) (*e.g.*, to randomize the size of each payload, and thus also randomize the size of each “*data chunk*”). This would have “*further secure[d] the plurality of data chunks,*” as claimed, which POSAs had reason to do to provide the benefit of additional security. *See, e.g.*, EX1047, [0056]-[0058], (and other exhibits cited in §VIII.G above, claim 4). As I also discussed above, to generate the random/ pseudorandom number used for that distribution, it would have been conventional to use an encryption key as a “seed” to a number generator. §VII.I above (citing corroboration for conventional number generators). Thus, POSAs had reason to use Foster’s KEK (an “*external key*”) or another conventional externally-received encryption key (*see* §VII.I above) as a seed to a number generator used to “*encrypt[] primary data with th[at] external key,*” as claimed.

K. Claim 8: The method of claim 1, wherein distributing the data set into a plurality of shares comprises: generating a split encryption key based on the external key; and distributing the data set based on the split encryption key.

240. As I discussed above (§VIII.G, Claim 4), POSAs had reason to generate Foster-Hayhurst’s content units by distributing the data set into payloads (*i.e.*, “*shares*”) according to a random/pseudorandom number. For example, as I

discussed, the random/pseudorandom number determines the size of each payload (the “*share*”) and/or determines what payload (the “*share*”) would have received a particular bit (byte or other portion) of the data set. *See* §VIII.G (citing corroboration). When that random/pseudorandom number is used to distribute the data set into payloads in one of the manners I discussed above, that random/pseudorandom number meets the claimed “*split encryption key*” because the sequence of number is the key used in the algorithm for distributing Foster-Hayhurst’s data into chunks, for the reasons I discussed above (§VIII.G). *See also* §VII.I (discussing the ’854 patent’s disclosure of split encryption keys at EX1001, 78:11-28).

241. As I also discussed in §VII.I with corroboration (EX1040, [0030]; EX1041, [0024], [0026]-[0027]), it would have been conventional to generate random/pseudo-random numbers using an encryption key – as a so-called “seed” for a number generator.



Because Foster’s KEK (the external key) is an encryption key used to secure the data set, POSAs would have had reason to also use Foster’s KEK as the seed to generate the above-discussed split encryption key.

242. As discussed above (§VIII.J, claim 7), POSAs had reason to use a conventional (external) encryption key to encrypt Foster-Hayhurst's data set. Thus, if the above-discussed random/pseudorandom sequence were not generated using Foster's KEK, POSAs had reason to use a conventional externally-received encryption key as the split-encryption-key (or the seed to generate that key), to achieve the well-known benefits that I discussed above that would have resulted from distributing data according to a random/pseudorandom number. My opinion in that regard is corroborated by EX1051, which discloses at [0037] that it was known that "some systems rely on a master key to generate derived keys, which are then used to encrypt data." Thus, it would have been conventional to use Foster's KEK as the master key to generate a split encryption key.

243. When using an additional key as or to generate the split encryption key, POSAs had reason for the key(s) to be stored and retrieved from Foster's database 426 (a storage system) because database 426 was already designed to store encryption keys, as I discussed above (§VIII.D.2, [1A]). Foster (EX1006), [0077] ("Database 426 may provide storage for information necessary to carry out the present invention such as a KMB, content usage conditions, a verifier, identifier information, etc."). Thus, a conventional encryption key would have been an alternative "external key" to use in Foster's encryption method and POSAs had reason to use that key as a split encryption key or as a seed to generate a split

encryption key, for the reasons I discussed above in this section. Accordingly, Foster-Hayhurst renders claim 8 obvious to POSAs.

L. Claim 9: The method of claim 8, further comprising encrypting each of the shares with the external key.

244. As I discussed above (§VIII.C), Foster-Hayhurst performs Foster’s encryption method. As I also discussed above (§VIII.D.7, [1C]), Foster discloses encrypting the title key of each payload (the “*share*”) with a KEK (the “*external key*”), which meets [1C]’s “*an encryption operation based on the external key to further secure the... data chunks*” under Patent Owner’s claim scope. *See also* EX1009, 8. Thus, by encrypting each payload’s (“*share’s*”) title key with a KEK (“*external key*”), Foster-Hayhurst is also “*encrypting each of the shares with the external key,*” as claimed, under Patent Owner’s assertion of the claim language’s scope. *See* §VIII.D.7 above ([1C]).

245. If Claim 9 requires performing an encryption operation directly to shares using the external key, POSAs had reason to encrypt each share with Foster’s KEK or an alternative (conventional) encryption key retrieved from an external storage system (like Foster’s database). Doing so would have provided additional security for each share beyond encrypting its title key, for reasons I discussed above with corroboration (§VII.B.7.b.i-§VII.B.7.b.iii).

246. As I also discussed above (§VIII.G, claim 4), POSAs had reason to distribute Foster-Hayhurst’s data into random/pseudorandom-sized chunks, whose

size would have been based on the external key. That distribution would have been “*encrypting each of the shares with the external key*,” as claimed.

M. Claim 10: The method of claim 9, wherein the data set is recoverable using at least a minimum number of the plurality of data chunks and the external key.

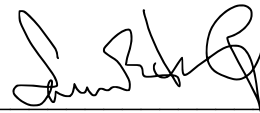
247. Foster-Hayhurst meets claim 10 for the same reason I discussed above. Foster-Hayhurst meets [1B] (§VIII.D.3) – which requires that the data set “*can be reconstructed using at least a minimum number of the plurality of data chunks*” – and claim 9 (§VIII.L) – which requires “*encrypting each of the shares with the external key*” (and thus conventionally would have required the external key to recover the data set).

* * *

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

I declare under penalty of perjury that the foregoing is true and correct.

Dated: November 22, 2023



Samrat Bhattacharjee

IX. APPENDIX: CLAIM LISTING

The following claim listing assigns element labels (*e.g.*, 1[PRE], [1A], etc.) to certain claims for convenience of reference.

Claim 1
1[PRE] A method for securely storing a data set, the method comprising:
[1A] receiving an external key from an external storage system,
[1B] generating a plurality of data chunks based on the data set, such that the data set can be reconstructed using at least a minimum number of the plurality of data chunks,
[1B1] wherein generating the data chunks comprises: distributing the data set into a plurality of shares, wherein each of the shares comprises less than all of the data set,
[1B2] [wherein generating the data chunks comprises:] accessing a plurality of distinct encryption keys,
[1B3] [wherein generating the data chunks comprises:] encrypting each of the shares with a respective one of the plurality of distinct encryption keys,
[1C] performing an encryption operation based on the external key to further secure the plurality of data chunks; and
[1D] storing with the plurality of data chunks data indicative of at least one of the distinct encryption keys on a plurality of different storage devices.
Claim 2
2. The method of claim 1, wherein storing with the plurality of data chunks data indicative of at least one of the distinct encryption keys comprises storing each of the plurality of data chunks with at least one of the distinct encryption keys.
Claim 3
3. The method of claim 1, wherein storing each of the plurality of data chunks with at least one of the distinct encryption keys comprises including with a particular data chunk of the plurality of data chunks data indicative of an encryption key that was used to encrypt a different data chunk of the plurality of data chunks.
Claim 4
4. The method of claim 1, wherein distributing the data set into a plurality of shares comprises using a random technique or pseudorandom technique.
Claim 5
5. The method of claim 1, wherein distributing the data set into a plurality of shares comprises using a deterministic technique or pseudorandom technique.

Claim 6
6. The method of claim 1, wherein each share of the plurality of shares comprises a substantially random distribution of a subset of the data set.
Claim 7
7. The method of claim 1, wherein performing an encryption operation based on the external key to further secure the plurality of data chunks comprises encrypting primary data with the external key.
Claim 8
8. The method of claim 1, wherein distributing the data set into a plurality of shares comprises: generating a split encryption key based on the external key; and distributing the data set based on the split encryption key.
Claim 9
9. The method of claim 8, further comprising encrypting each of the shares with the external key.
Claim 10
10. The method of claim 9, wherein the data set is recoverable using at least a minimum number of the plurality of data chunks and the external key.