

Filed on behalf of Google LLC by:

Paper No. ____

Elisabeth H. Hunt, Reg. No. 67,336

Gregory S. Nieberg, Reg. No. 57,063

Nathan R. Speed (*pro hac vice* application forthcoming)

WOLF, GREENFIELD & SACKS, P.C.

600 Atlantic Avenue

Boston, MA 02210

(617) 646-8000 Phone

(617) 646-8646 Fax

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

**PETITION FOR INTER PARTES REVIEW
UNDER 35 U.S.C. §§311-319 AND 37 C.F.R. §42.1 et seq**

TABLE OF CONTENTS

MANDATORY NOTICES.....	x
A. Real Party-In-Interest	x
B. Related Matters	x
1. United States Patent & Trademark Office.....	x
2. U.S. District Court for the Eastern District of Virginia.....	xi
C. Counsel and Service Information - §42.8(b)(3) and (4)	xi
I. INTRODUCTION	1
II. STANDING	3
III. UNPATENTABILITY GROUNDS.....	3
IV. THE '854 PATENT.....	4
A. Specification.....	4
B. Person of Ordinary Skill in the Art (“POSA”).....	6
C. Prosecution History	6
D. Challenged Claims	6
V. CLAIM INTERPRETATION	7
VI. GROUND 1: CLAIMS 1-10 ARE ANTICIPATED OR ALTERNATIVELY OBVIOUS OVER ORSINI.....	8
A. Orsini (EX1005).....	8
B. Claim 1:	16
1. 1[PRE] A method for securely storing a data set, the method comprising:	16
2. [1A] receiving an external key from an external storage system, ...	16
a. Orsini discloses [1A].....	17
i. Encryption keys received from and stored at Orsini’s “external” key management.....	17
ii. Public-key encryption keys received from a certificate authority.....	22
b. Alternatively, Orsini renders obvious [1A]	24

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	29
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, ...	30
5.	[1B2] [wherein generating the data chunks comprises:] accessing a plurality of distinct encryption keys, ...	31
6.	[1B3] [wherein generating the data chunks comprises:] encrypting each of the shares with a respective one of the plurality of distinct encryption keys	32
7.	[1C] performing an encryption operation based on the external key to further secure the plurality of data chunks; and	32
a.	Orsini discloses [1C]	33
i.	Performing encryption steps in different order	33
ii.	Repeating encryption steps.	35
iii.	Encrypting with a public key.	38
b.	Alternatively, [1C] is obvious over Orsini	39
i.	POSAs had reason to re-encrypt data chunks with an external key	39
ii.	POSAs had reason to re-encrypt data chunks with an externally-received group key.	40
iii.	POSAs had reason to reorder data within each chunk based on an external key	41
iv.	POSAs had reason to encrypt [1B3]’s distinct encryption keys with an externally-stored key-encryption key	43
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.	45
C.	Claim 2: [1D]’s storing...comprises storing each of the plurality of data chunks with at least one of the distinct encryption keys	48
D.	Claim 3: [1D]’s storing...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.	48
E. Claim 4: [1B1]’s distributing...comprises using a random technique or pseudorandom technique.....	48
F. Claim 5: [1B1]’s distributing...comprises using a deterministic technique or pseudorandom technique.....	49
G. Claim 6: ...each share...comprises a substantially random distribution of a subset of the data set.....	50
H. Claim 7: [1C]’s encryption operation...comprises encrypting primary data with the external key.....	50
I. Claim 8: [1B1]’s distributing...comprises: generating a split encryption key based on the external key; and distributing the data set based on the split encryption key.	51
J. Claim 9: The method of claim 8, further comprising encrypting each of the shares with the external key.	53
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.	54
VII. GROUND 2: CLAIMS 1-10 WOULD HAVE BEEN OBVIOUS OVER FOSTER-HAYHURST	55
A. Foster (EX1006).....	55
B. Hayhurst (EX1007)	59
C. Foster-Hayhurst.....	61
D. Claim 1:	65
1. 1[PRE]	65
2. [1A].....	67
3. [1B].....	71
4. [1B1]	74
5. [1B2]	75
6. [1B3]	77
7. [1C]	77
8. [1D].....	80

E. Claim 2	82
F. Claim 3	83
G. Claim 4	84
H. Claim 5	85
I. Claim 6	85
J. Claim 7	86
K. Claim 8	87
L. Claim 9	88
M. Claim 10	89
VIII. NO BASIS EXISTS FOR DISCRETIONARY DENIAL	89
A. Section 314(a)	89
1. Stay Potential	90
2. Trial Timing.....	90
3. Litigation Investment.....	90
4. Issue Overlap	91
5. Litigation Defendants	91
6. Other Circumstances.....	91
B. Section 325(d)	91
1. Step 1: Art and Arguments Not Previously Considered.....	91
2. Step 2: The Office Erred Materially	92
a. Factor (c): Orsini Was Not Previously Evaluated	92
b. Factors (e)-(f): The Office Erred.....	92
IX. CONCLUSION.....	93
X. APPENDIX: CLAIM LISTING.....	95

TABLE OF AUTHORITIES

Cases

<i>10X Genomics, Inc. v. Bio-Rad Labs.</i> , IPR2020-00086, Paper 8 (Apr. 27, 2020).....	8
<i>Advanced Bionics v. Med-El</i> , IPR2019-01469, Paper 6 (Feb. 13, 2020).....	91, 92
<i>Apple Inc. v. Fintiv Inc.</i> , IPR2020-00019, Paper 11 (Mar. 20, 2020)	89, 90
<i>Becton, Dickinson & Co. v. B. Braun Melsungen AG</i> , IPR2017-01586, Paper 8 (Dec. 15, 2017).....	92
<i>Dish Network LLC v. Broadband iTV, Inc.</i> , IPR2020-01332, Paper 14 (Jan. 27, 2021).....	90
<i>DJI Europe BV v. Textron Innovations, Inc.</i> , IPR2022-00453, Paper 10 (Aug. 3, 2022)	92
<i>Intel Corp. v. PACT XPP Schweiz AG</i> , 61 F.4th th 1373 (Fed. Cir. 2023).....	40, 63
<i>KSR Int’l Co. v. Teleflex Inc.</i> , 550 U.S. 398 (2007).....	27, 41, 63, 83
<i>Markforged Inc. v. Continuous Composites</i> , IPR2022-00679, Paper 7 (Oct. 25, 2022)	91
<i>Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.</i> , 868 F.3d 1013 (Fed. Cir. 2017)	7
<i>PEAG LLC d/b/a JLab Audio v. VARTA Microbattery GmbH</i> , IPR2020-01214, Paper 8 (Jan. 6, 2021).....	90
<i>Sand Revolution II, LLC, v. Cont’l Intermodal Group–Trucking</i> , IPR2019-01393, Paper 24 (June 16, 2020).....	90, 91
<i>ShenZhen Apaltek Co., Ltd. v. Asetek Danmark A/S</i> , IPR2022-01317, Paper 7 (Feb. 6, 2023).....	92

<i>Uber Techs., Inc. v. X One, Inc.</i> , 957 F.3d 1334 (Fed. Cir. 2020)	28
---	----

Statutes

35 U.S.C. §102.....	3
35 U.S.C. §103.....	3
35 U.S.C. §314(a)	89
35 U.S.C. §325(d)	91, 92

Regulations

37 C.F.R. §42.100(b)	7
37 C.F.R. §42.104(a).....	3

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. (“Bhattacharjee”)
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”)
1008	RESERVED
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

Exhibit	Description
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
1038	U.S. Patent Application Publication No. 2004/0177257
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
1042	RESERVED
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

Exhibit	Description
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
1057	RESERVED
1058	RESERVED
1059	RESERVED
1060	RESERVED
1061	U.S. Patent Application Publication No. 2006/0059347
1062	U.S. Patent Application Publication No. 2006/0064386
1063	Rule 16(b) Scheduling Order in <i>Security First Innovations, LLC v. Google, LLC</i> , 2:23-cv-00097-JWK-RJK, Dkt. 54 (July 20, 2023)
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

MANDATORY NOTICES

A. Real Party-In-Interest

Petitioner Google LLC is the Real Party-in-Interest.¹

B. Related Matters

1. United States Patent & Trademark Office

The application from which U.S. Patent No. 10,452,854 issued is a Continuation of U.S. Patent Application No. 14/546,887 (U.S. Patent No. 10,108,807), filed on November 18, 2014, which is a Continuation of U.S. Patent Application No. 14/113,903 (U.S. Patent No. 9,317,705), filed December 19, 2013, which is a Continuation of U.S. Patent Application No. 13/668,433 (U.S. Patent No. 8,644,502), filed November 5, 2012, which is a Continuation of U.S. Patent Application No. 13/172,682 (U.S. Patent No. 8,320,560), filed June 29, 2011, which is a Continuation of U.S. Patent Application No. 11/602,667 (U.S. Patent No. 8,009,830), filed November 20, 2006, which claims priority to U.S. Provisional Application No. 60/738,231, filed November 18, 2005.

The following U.S. patent applications claim the benefit of priority to U.S. Patent No. 10,452,854:

¹ Google LLC is a subsidiary of XXVI Holdings Inc., which is a subsidiary of Alphabet Inc. XXVI Holdings Inc. and Alphabet Inc. are not real parties-in-interest to this proceeding.

(i) U.S. Patent Application No. 16/658,506 (U.S. Patent No. 11,068,609), filed October 21, 2019; (ii) U.S. Patent Application No. 17/347,268, filed June 14, 2021.

2. U.S. District Court for the Eastern District of Virginia

(i) *Security First Innovations, LLC v. Google LLC f/k/a Google Inc.*, Case No. 1:23-cv-00329, (ii) *Security First Innovations, LLC v. Google LLC*, Case No. 2:23-cv-00097.

C. Counsel and Service Information - §42.8(b)(3) and (4)

Lead Counsel	Elisabeth H. Hunt, Reg. No. 67,336
Backup Counsel	Gregory S. Nieberg, Reg. No. 57,063 Nathan R. Speed (<i>pro hac vice</i> application forthcoming)
Service Information	<u>E-mail</u> : EHunt-PTAB@wolfgreenfield.com GNieberg-PTAB@wolfgreenfield.com nspeed@wolfgreenfield.com <u>Post and hand delivery</u> : Wolf, Greenfield & Sacks, P.C. 600 Atlantic Avenue Boston, MA 02210-2206 <u>Telephone</u> : 617-646-8000 <u>Facsimile</u> : 617-646-8646

A power of attorney is submitted with the Petition. Counsel for Petitioner consents to service of all documents via electronic mail.

I. INTRODUCTION

Petitioner requests *inter partes* review of claims 1-10 (“the challenged claims”) of U.S. Patent No. 10,452,854 (“the ’854 patent”) (EX1001).

The ’854 patent discusses methods for “securely storing” 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* 2:45-54. The encrypted portions of data can then be stored in different locations. EX1001, 2:56-58.

To perform encryption, the patent relies on conventional prior-art “encryption methods or algorithms” that were published years before the patent’s earliest alleged priority date. EX1001, 61:5-46. It likewise relies on conventional techniques for splitting data into portions.

Beyond conventional encryption and data-splitting operations, the challenged method claims use an “external server” to store an “external key” that is used to perform a post-encryption step that encrypts the data again after it has already been split and encrypted a first time. But re-encrypting data and using an external key to do so were conventional techniques known in the prior art before the purported invention. Given the state of the art, the challenged claims should never have issued. EX1003 (hereinafter “Bhattacharjee”), ¶¶29-30.

Ground 1 demonstrates that Orsini (EX1005) anticipates and/or renders obvious claims 1-10. Orsini is Patent Owner’s (“SFI’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. Orsini also explicitly teaches 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. The challenged claims are unpatentable over Orsini. *Bhattacharjee*, ¶31.

Ground 2 demonstrates that claims 1-10 are also unpatentable over Foster (EX1006) combined with Hayhurst (EX1007). Foster discloses splitting data into multiple portions, encrypting those portions with individual encryption keys, encrypting those individual keys with a common encryption key received from a storage system, and transmitting the encrypted portions and keys for subsequent storage. Hayhurst discloses techniques for storing encrypted portions of data—like those Foster generates—on a plurality of storage devices. Persons of ordinary skill in the art (“POSAs”) 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. *Bhattacharjee*, ¶32.

II. STANDING

Petitioner certifies that the '854 patent is available for IPR and that Petitioner is not barred or estopped from requesting IPR as to the challenged claims. 37 C.F.R. §42.104(a).

III. UNPATENTABILITY GROUNDS

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	§§102/103
2	Foster (EX1006), Hayhurst (EX1007)	1-10	§103

Orsini, Foster, and Hayhurst are each prior art under pre-AIA 35 U.S.C. §102(b) at least because each published more than one year before the '854 patent's earliest alleged priority date of November 18, 2005. Orsini, Foster, and Hayhurst were not cited during prosecution. The Orsini patent application issued as U.S. Patent No. 7,391,865, which was listed in an IDS but was not used to reject any claim.

IV. THE '854 PATENT²

A. Specification

The '854 patent relates to techniques for securing electronic data from unauthorized access. 1:25-26. By 2005, individuals and businesses were storing and transmitting an ever-increasing amount of sensitive information over computer networks. 1:30-37. According to the 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.” 1:47-51. The 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. 1:60-62. This was allegedly problematic because users often stored keys in unsecure systems or lost access to devices storing their keys (*e.g.*, mobile phones). 1:63-2:13; Bhattacharjee, ¶33.

To overcome this alleged problem, the patent discusses a system for securing data that is purportedly “user-independent” (2:42-44) because it relies on “one or more secure servers” that users can access but from which “actual [encryption] keys and other authentication data” is not released (8:66-9:6). In other words, the

² Unless otherwise noted, Section IV’s citations are to EX1001.

encryption keys remain secured on the cryptographic servers and are not given to users. 8:66-9:6. Allegedly, “[t]his server-centric storage of keys...provides for user-independent security, portability, availability, and straightforwardness.” 9:6-8; Bhattacharjee, ¶34.

The 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 the parsing/splitting/separating), and **[3]** storing the parsed/split/separated “data that has been encrypted” in one or more “locations.” 2:45-58. The patent says that steps **[1]** and **[2]** can be “repeated” to parse/split/separate and/or encrypt data multiple times. 2:52-55; Bhattacharjee, ¶35.

As seen *infra* §IV.C, the challenged method claims recite performing the patent’s three basic encryption steps and further recite using an “external [encryption] key” to perform an additional “encryption operation” to further secure the data set’s “data chunks.” The patent describes an external key as one that is “provided by a user or an external system.” 77:59-61. This “external key” can be used to “post encrypt” data after the data set has first been parsed and encrypted. 79:15-19; FIG. 36 (step 3619). The patent ascribes no advantages to using “external” keys or to receiving such keys from “external” systems. Bhattacharjee, ¶36.

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

A POSA at the time of the ’854 patent’s November 18, 2005 earliest claimed priority date would have had at least a bachelor’s degree in computer science, computer engineering, or a related field, with three years of experience in the area of securing data from unauthorized access or use. A higher level of education may substitute for less experience. Bhattacharjee, ¶¶37-38.

C. Prosecution History

The challenged claims were allowed without any rejections. EX1002, 33-38; Bhattacharjee, ¶42. As discussed *infra* §VIII.B, U.S. Patent No. 7,391,865 (the issued patent version of Orsini) was included in an IDS (EX1002, 45) but was never cited as a basis for rejecting the challenged claims.

D. Challenged Claims

Claims 1-10 are reproduced in the Claim Listing (*infra* §X), with labeled limitations (*e.g.*, 1[**PRE**]-[**1D**]).

As discussed *infra* (§§VI.B.3-VI.B.6, §§VII.D.3-VII.D.6), 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**] (§VI.B.2, §VII.D.2) and [**1C**] (§VI.B.7, §VII.D.7) use an external key to “perform[] an encryption operation” that “further secure[s]” the data chunks, and limitation [**1D**] (§VI.B.8, §VII.D.8) stores the data chunks on different storage devices and stores

with each data chunk “data indicative” of at least one of [1B]’s “distinct encryption keys.” As Grounds 1-2 demonstrate, claim 1’s steps (a) are disclosed in and/or would have been obvious over Orsini and (b) would have been obvious over the combination of Foster and Hayhurst. Bhattacharjee, ¶¶43-44.

The dependent claims recite conventional data-encryption features. For example, dependent claims 4-6 recite 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 external key, distributing data based on that split key, and encrypting and recovering data shares based on the external key. 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 like Orsini and Foster. Bhattacharjee, ¶45.

V. CLAIM INTERPRETATION

Claim terms are construed herein according to their ordinary and customary meaning in light of the specification and prosecution history. 37 C.F.R. §42.100(b). The Board “need only construe terms that are in controversy, and only to the extent necessary to resolve the controversy” regarding unpatentability. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017) (cleaned up). In the concurrent district-court proceeding, the parties have served competing proposed constructions of various claim terms. EX1064-EX1065. As

discussed in detail where relevant within the Grounds below, this Petition presents alternative mappings of the prior-art disclosures to both parties' proposed constructions of disputed terms, thus demonstrating unpatentability regardless of which proposed construction is correct. *See, e.g., 10X Genomics, Inc. v. Bio-Rad Labs.*, IPR2020-00086, Paper 8, 18-22 (Apr. 27, 2020) ("Petitioner proposes alternative claim constructions and presents at least one ground of unpatentability for each construction.").

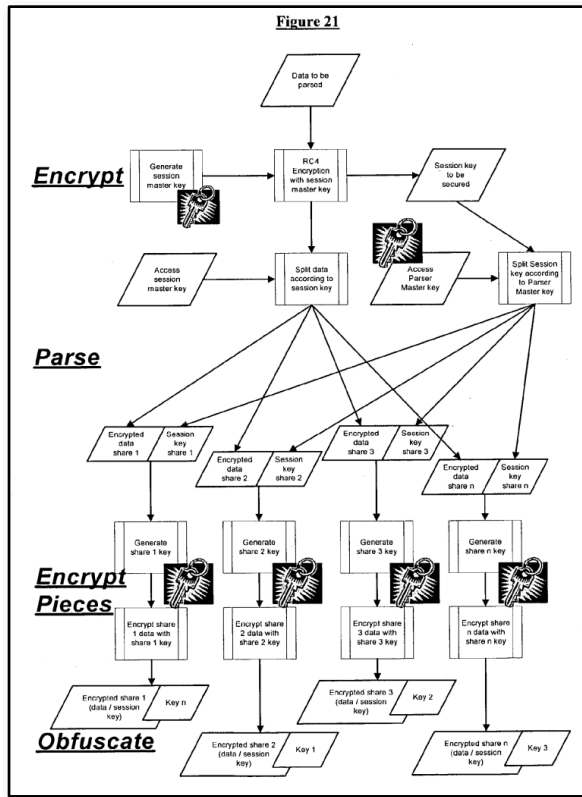
VI. GROUND 1: CLAIMS 1-10 ARE ANTICIPATED OR ALTERNATIVELY OBVIOUS OVER ORSINI³

A. Orsini (EX1005)

The '854 patent and Orsini share an assignee and overlapping inventors, and significant portions of their disclosures are substantively identical. *Compare* Orsini, [0001]-[0423], FIGS. 1-25 *with* EX1001, 1:1-66:3, FIGS. 1-25; *see also* EX1010 (redline comparison); Bhattacharjee, ¶¶46-48.

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

³ Unless otherwise noted, Section VI's citations are to Orsini.



Orsini

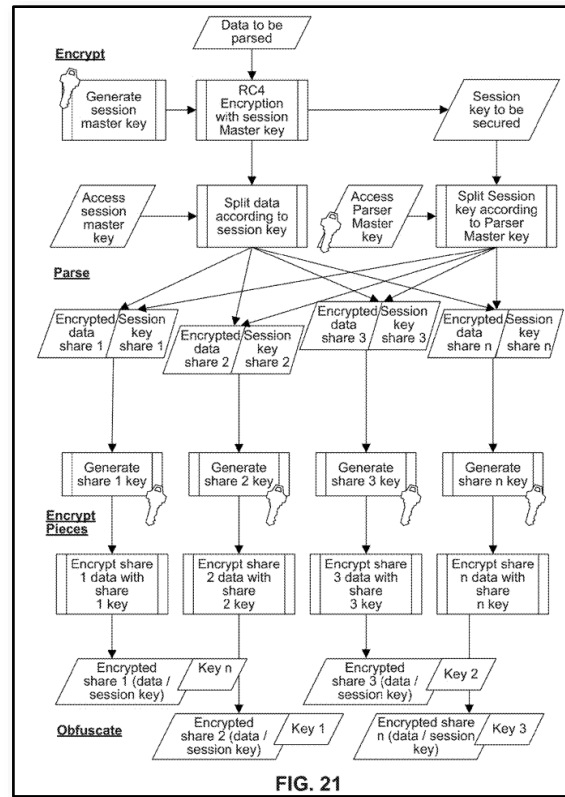
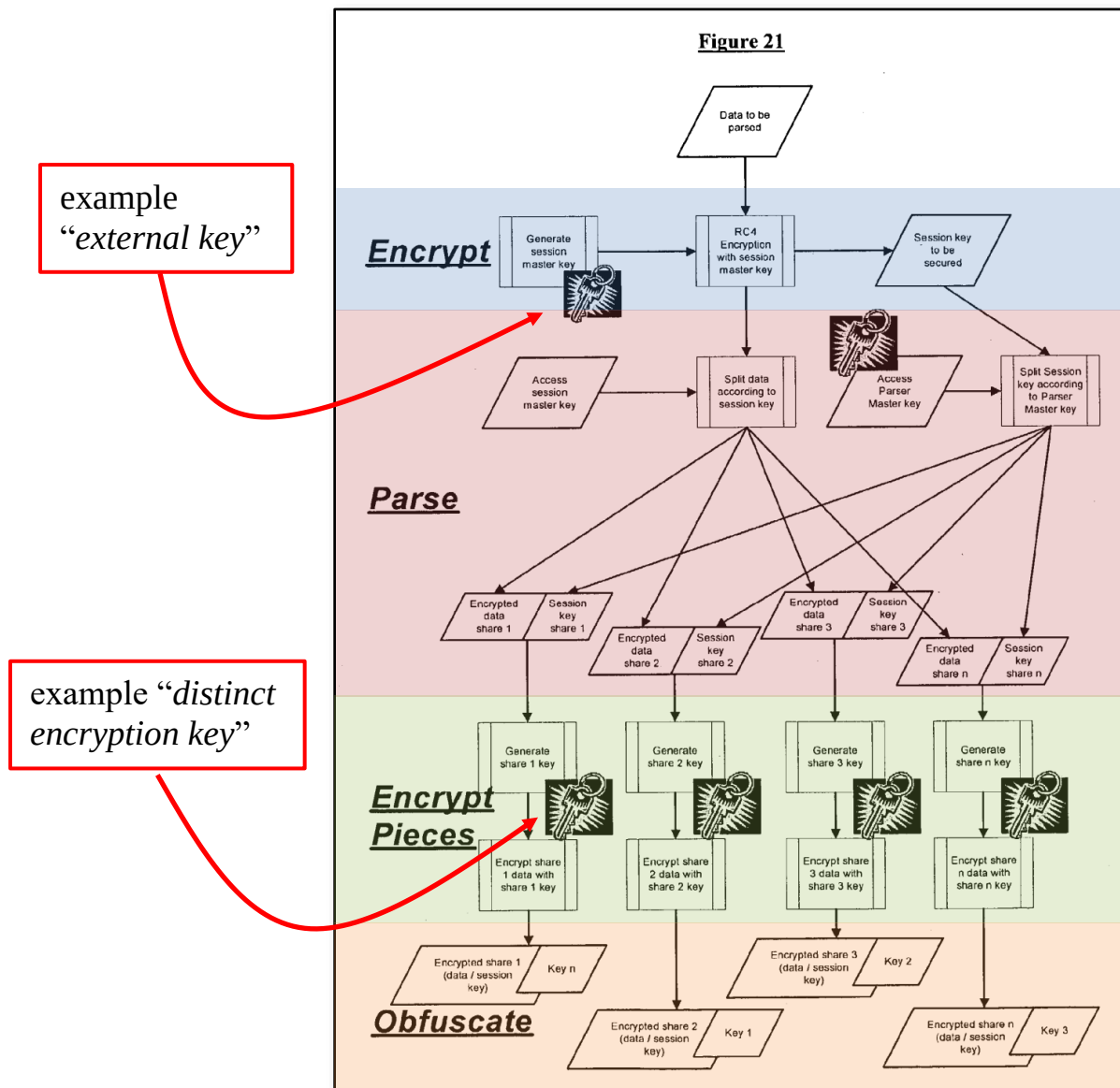


FIG. 21

'854 Patent

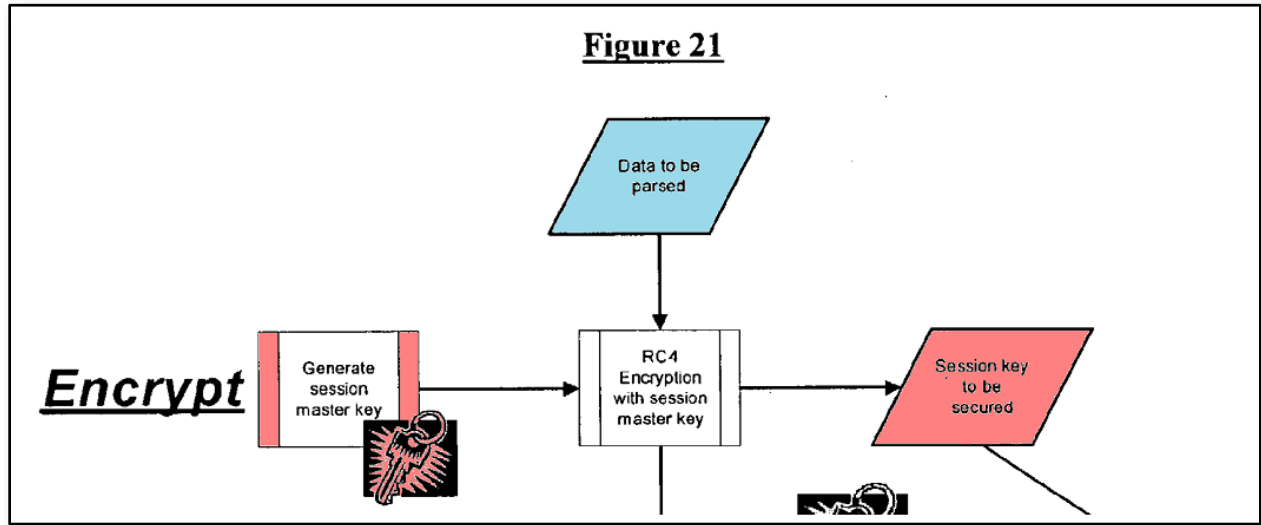
Bhattacharjee, ¶¶49-50.

Figure 21's exemplary method (color-annotated below) 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,” *infra* §VI.B.2.a) and multiple “share keys” (which meet limitation [1B2]’s “distinct encryption keys,” *infra* §VI.B.5):



Bhattacharjee, ¶¶50-51.

In Orsini’s “encrypt” stage (color-annotated below), a data set (blue) is encrypted with a “session master key” (red) ([0294]-[0295]),⁴ using any “well known” encryption algorithm:

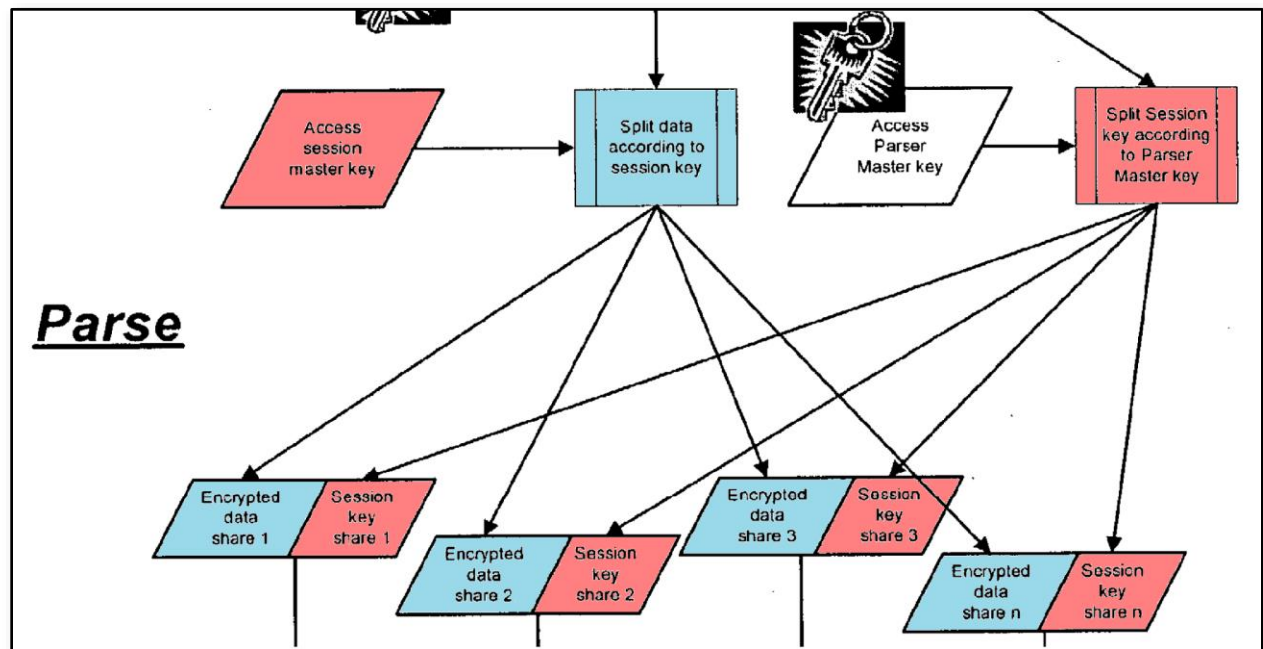


See also [0335] (“A wide variety of encryption methodologies are suitable for use in the methods of the present invention....”), [0371], [0378]. Orsini further discloses “external session key management” ([0293]), which POSAs understood includes generating and storing the session keys externally. Bhattacharjee, ¶52.

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.” [0296]. The session master key itself can be split “according to the pattern of the Parser Master Key,” which is yet another encryption key. [0297], [0391]-[0392]. Each portion of

⁴ “Master keys” are used to derive other encryption keys. Bhattacharjee, ¶52.

the split encrypted data (blue “data shares” below) is then paired with a portion of the split session key (red “key shares” below):

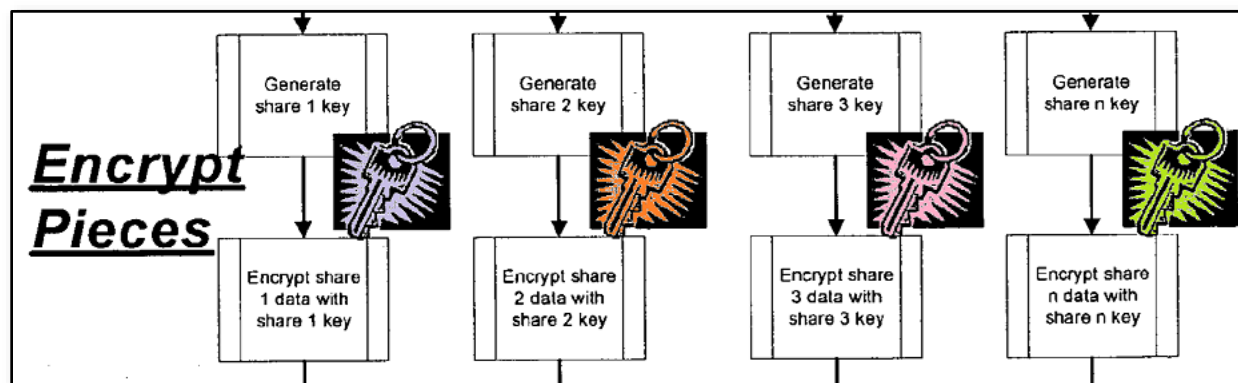


Bhattacharjee, ¶53.

In the “encrypt pieces” stage, each share pair (each blue-red combination of an encrypted data share with a session key share above) is encrypted with its own distinct encryption “share key” (purple, orange, pink, and green below), creating an encrypted share (which meets [1B]’s “*data chunk*,” §VI.B.3 *infra*). Each encrypted share includes a portion of the (encrypted) original data set and a portion of the

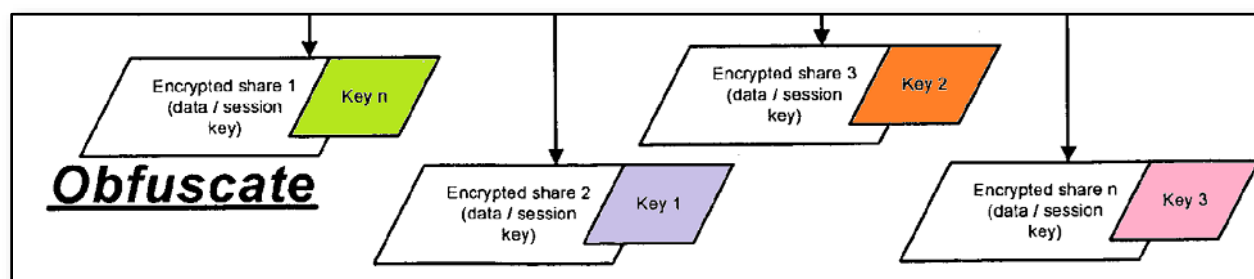
original session master key. [0298]-[0299], [0395]-[0396]. Bhattacharjee, ¶54.

This is depicted below:



Finally, in the “obfuscate” stage, each encrypted share is paired with a share key that encrypted a *different* share (e.g., encrypted share 2 is paired with share 1 key, etc.). [0299].⁵ The different pairs of encrypted share and share key can then be stored, e.g., in different data depositories. [0299], [0379]. Bhattacharjee, ¶55.

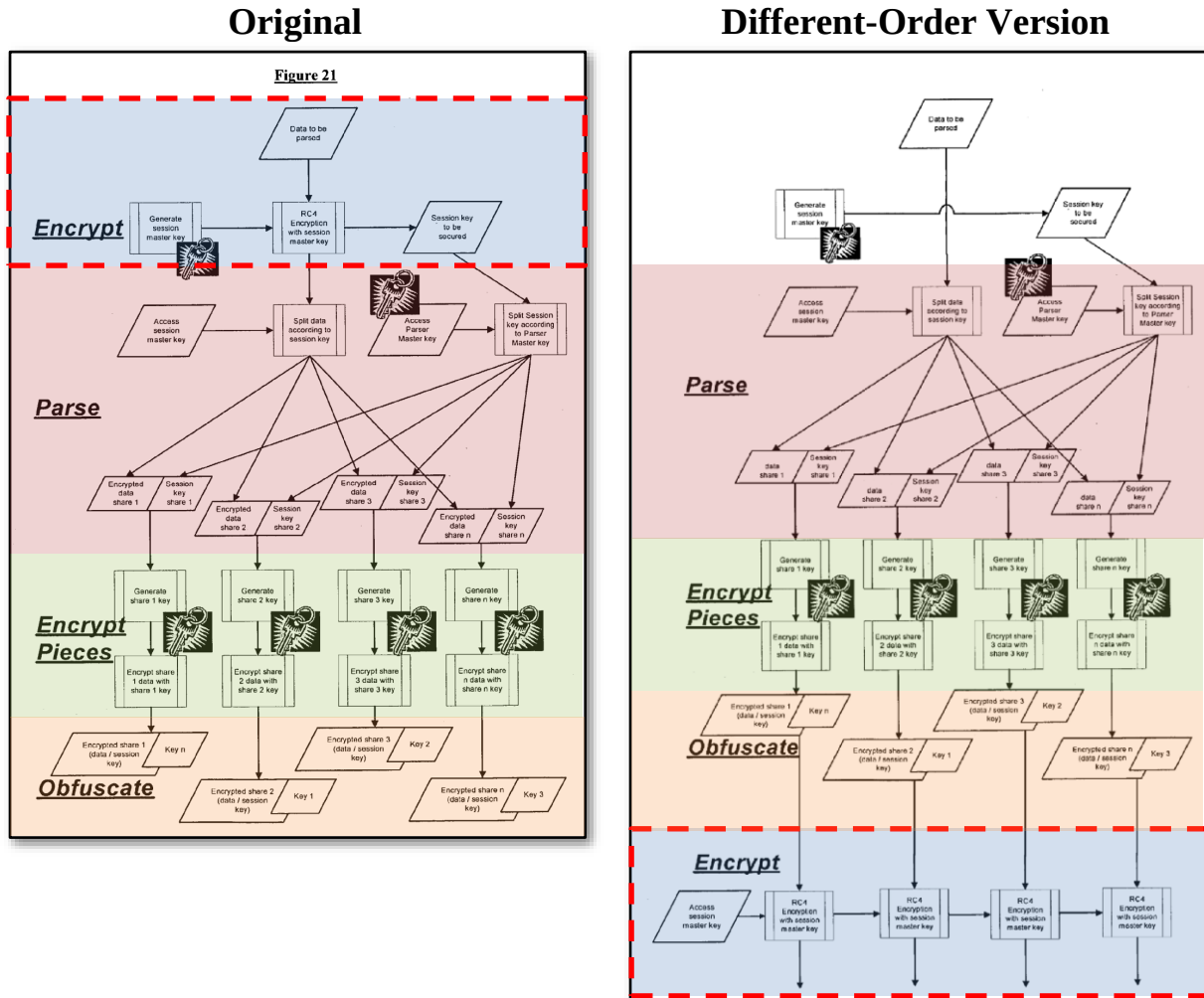
This is depicted below:



⁵ All emphases herein are added unless otherwise indicated.

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 (§VI.B *infra*) 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." [0301].

One exemplary different-order Orsini embodiment that meets claim 1 (as discussed §VI.B.7.a.i *infra*) is conceptually illustrated below (right), where the “encrypt” step is reordered to be the last step before storing encrypted data:



Bhattacharjee, ¶¶56-59.

B. Claim 1:

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

Orsini discloses “a method for *securing* virtually *any type of data*,” and a step in the method is “*storing...data that has been encrypted*.” [0013]; Abstract (“*securing* sensitive *data*”). Indeed, Orsini’s claims recite methods for “encrypting a *data set*” and then “storing” the encrypted data. *See, e.g.*, claims 1, 10, 20, 30. Storing data in an encrypted state is a way of securely storing data. Bhattacharjee, ¶¶60-61.

In an exemplary embodiment, Orsini’s method is executed by “a parser program” that performs “encryption” “on any *data set*.” [0278]; *see also* [0013]; Bhattacharjee, ¶62. Although the parties dispute the definition of “data set” (EX1064, 6-7; EX1065, 9-10), Orsini’s use of the identical term ([0278]) in a specification that shares disclosure and inventors with the ’854 patent meets that claim language regardless of which party’s definition is adopted.

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

As discussed below, Orsini discloses receiving an “*external key*” from an “*external storage system*” under both parties’ constructions of those terms. Alternatively, in view of Orsini’s disclosure and POSAs’ general knowledge of encryption and storage techniques, POSAs had reason to implement Orsini’s

encryption method to include limitation [1A] under both parties' constructions. Bhattacharjee, ¶63.

a. Orsini discloses [1A]

As discussed below, Orsini's encryption methods use "external session key management" and public encryption keys. Orsini's "external session key management" meets [1A] under SFI's proposed construction, and Orsini's use of public encryption keys meets [1A] under both parties' proposed constructions.

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

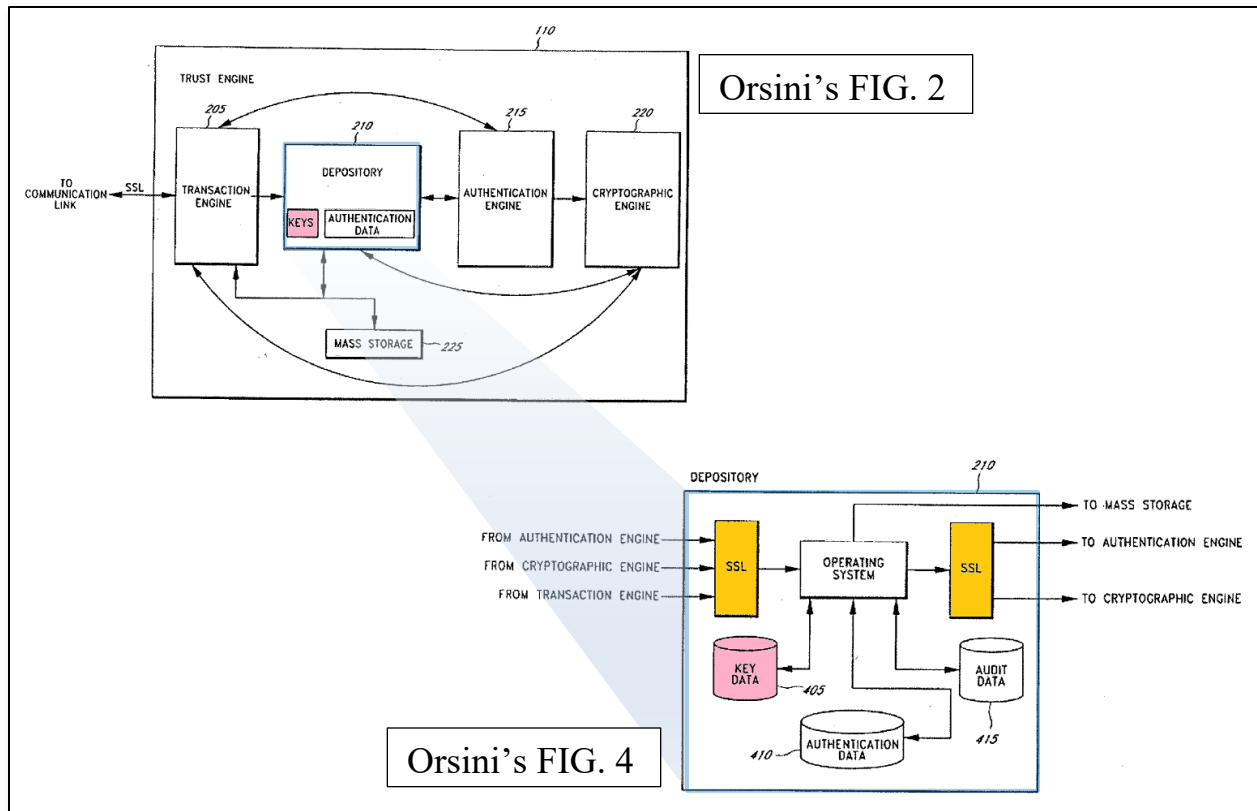
SFI argues that "[n]o construction [is] necessary" for [1A]'s "*external key*," or in the alternative construes it as "a key stored separately from the data chunks." EX1065, 8-9. As discussed below, Orsini meets [1A] under either alternative. Bhattacharjee, ¶¶64-65.

Orsini's Figure 21 shows its parser software generating the session master key without specifying at what physical device the generating occurs. Even if some embodiments generate the session master key at the encryption device rather than receiving it from a storage system, Orsini discloses embodiments where its encryption techniques "may be implemented with ***external session key management***." [0293], *see also* [0329] ("passing the Session Master Key...to the calling program for ***external management***"), [0381]-[0386], [0391]-[0394], [0399]-[0402], FIGS. 1-7. POSAs understood that "[k]ey [m]anagement" generally refers

to “[t]he various processes that deal with the **creation**, distribution, authentication, and **storage** of keys.” EX1015, [0030], [0044]. Bhattacharjee, ¶¶66-67. Thus, POSAs understood that instead of the session master key being internally generated, it can be created, distributed, and stored by an external session key management for later use by the system implementing Orsini’s encryption methods. Bhattacharjee, ¶67.

Orsini illustrates an exemplary storage system for encryption keys as “depository 210” in Figures 2 and 4 (annotated below). In Figure 2, cryptographic engine 220 and depository 210 are part of “trust engine 110,” which “comprises one or more secure servers.” [0034], [0036], [0066]; *see also* [0063]-[0113]. Orsini’s encryption method discussed below (§§VI.B.3-VI.B.8) is implemented in cryptographic engine 220. [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”); Bhattacharjee, ¶68.

Depository 210—illustrated in more detail in Figure 4 (annotated below right)—“stores cryptographic keys”; specifically, “[key] data 405” (pink below) which “correspond[s]” to the cryptographic keys. [0085].



POSAs would have understood from these disclosures that encryption keys used by cryptographic engine 220 (e.g., the session master key discussed *supra* §VI.A) are stored in and received from depository 210 (e.g., as key data 405). Bhattarcharjee, ¶69.

POSAs would have further understood from these disclosures that cryptographic engine 220 and depository 210 are stand-alone components implemented on different “secure servers” ([0066], [0079]) and that the depository is not built into the cryptographic engine. Bhattarcharjee, ¶70. In particular, Figure 4 depicts the cryptographic engine communicating with the depository using SSL (Secure Sockets Layer), a network-based security protocol. [0079], [0095];

EX1011, 5:19-45, 16:43-45. The cryptographic engine and depository communicate “using conventional HTTP routing techniques” and “URLs” (Uniform Resource Locators). [0081]. POSAs understood these protocols are conventionally used for communication between systems separated by different networks and/or the Internet. Bhattacharjee, ¶¶71-72 (citing, *e.g.*, EX1012, [0052], [0065]; EX1013, 3:16-29).

POSAs understood from Orsini’s disclosure that to perform a cryptographic operation when keys are externally managed, the cryptographic engine receives encryption keys from depository 210’s key-storage system. [0090] (“cryptographic keys are generated at the cryptographic engine 220 and forwarded to the depository 210”); [0095] (depository 210 in FIG. 4 “stores data 405 corresponding to the cryptographic keys”); [0107] (“cryptographic engine 220 also comprises a cryptographic handling module 625”); [0134] (“cryptographic handling module 625...*request[s] the matching private key be assembled from...depository 210*” and “use[s] the matching private key to perform cryptographic actions”); Bhattacharjee, ¶73. 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”). Bhattacharjee, ¶73. *See also infra* §VI.B.2.b (demonstrating receiving encryption keys from external storage system was also obvious over Orsini).

Under SFI's view that [1A]'s "*external key*" requires no construction and SFI'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") is an "*external key*." Bhattacharjee, ¶74.

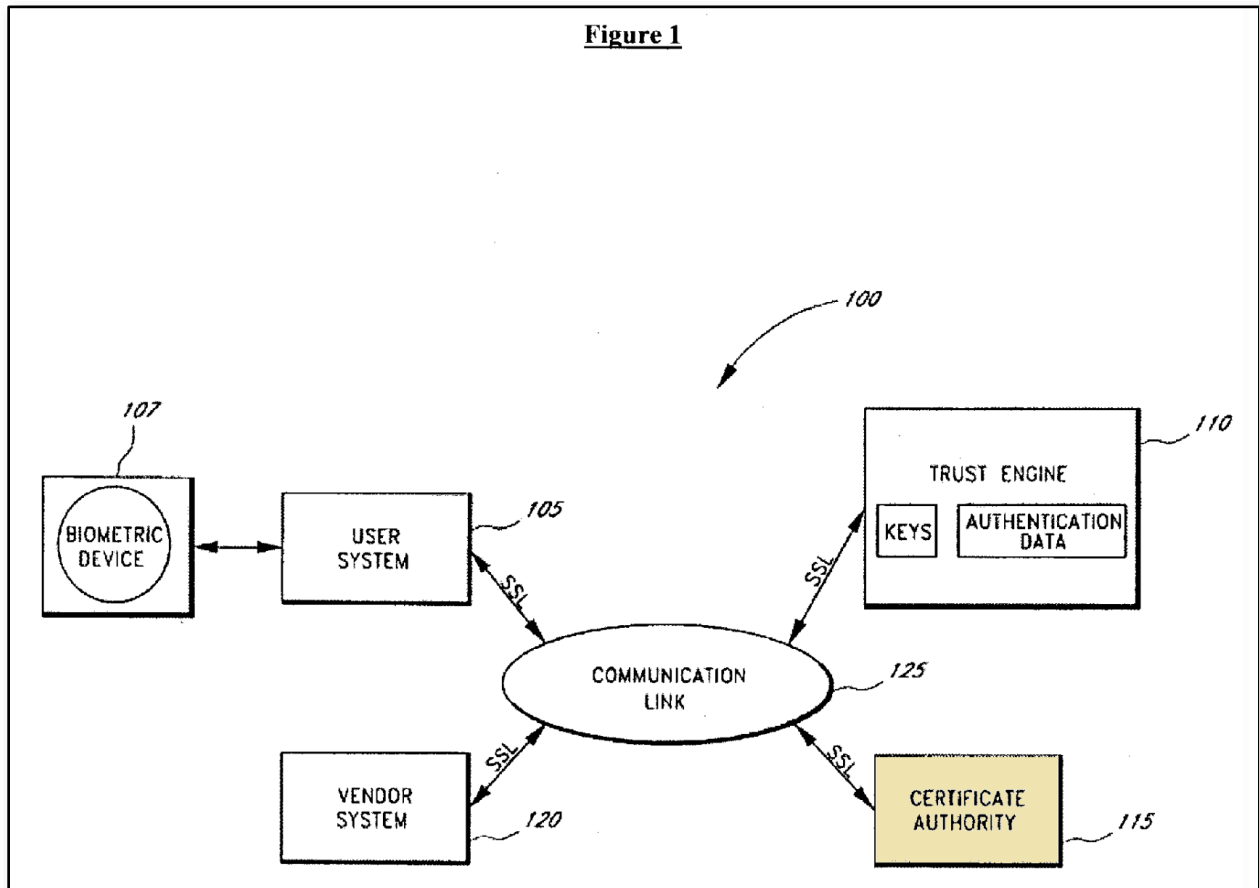
Under SFI's alternative construction of "*external key*" as "a key stored separately from the data chunks," (EX1065, 8), POSAs understood Orsini's session master key is stored separately from "*data chunks*" (encrypted shares, as discussed §VI.B.3 *infra*) because (in some Orsini embodiments) the session master key can be requested from depository 210 and used by the cryptographic engine ([0134] **before** the encrypted shares are generated, as discussed §VI.A *supra*. Bhattacharjee, ¶¶75-76.

Orsini also discloses that embodiments can "stor[e] the encryption master key data **separately from the data**" encrypted with that master key ([0055], [0057]), *e.g.*, as in Orsini's FIG. 22. *See also* [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 Ground 1 demonstrates, because the session master key's storage location does not affect the method steps' mapping to the other claim limitations. Bhattacharjee, ¶77.

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

Alternatively, Orsini's use of public encryption keys received from certificate authority 115 meets [1A] under both parties' proposed constructions. "[C]ertificate authority 115 generally issues digital certificates," which "include[] a public key of the user or system." [0125], [0007]. Bhattacharjee, ¶¶78-79.

Orsini's Figure 1 (color-annotated below) illustrates certificate authority 115 communicating with trust engine 110 over, *e.g.*, the Internet. [0062], [0076]-[0078], [0083].



The certificate authority is “**external** to the trust engine.” [0130]. As Figure 2 illustrates, trust engine 110 includes cryptographic engine 220. Thus, the certificate authority is also “**external**” to the cryptographic engine, which executes Orsini’s encryption method, as discussed above (§VI.B.2.a.i). A public key received from the certificate authority therefore meets the interpretation of “*external key*” in SFI’s infringement contentions (*see supra* §VI.B.2.a.i). Bhattacharjee, ¶80.

For the same reason, the certificate authority is “*external*” to the storage system(s) within the trust engine that store “*data chunks*” (*see* §VI.B.8 *infra*). For example, as shown in Figure 1 above, the trust engine processes “authentication data,” which is stored in depository 210. [0095], FIG. 2 (trust engine 110 includes depository 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 SFI’s alternative construction of “*external key*” because it is “a key stored separately from the data chunks.” EX1065, 8. Bhattacharjee, ¶81.

The certificate authority is controlled by “a trusted **third-party** organization or company.” [0072]. Certificate authorities “also issue public-private cryptographic key pairs” (EX1067, [0004]), which POSAs understood included generating these keys. *See also* EX1067, [0016] (describing modules that

“generate[]”and “securely store[] and/or distribute[]” cryptographic keys). Thus, a public key generated by and received from the certificate authority meet’s Google’s construction of “*external key*” because it is “a key generated by a third party.” EX1064, 11-12. Bhattacharjee, ¶80.

POSAs would have understood from Orsini’s disclosure that the certificate authority stores public keys in a storage system so that the certificate authority can distribute a public key upon request. [0128] (noting third parties may “request[] the user’s digital certificate,” which, as discussed above, comprises the user’s public key). Alternatively, it would have been obvious for the certificate authority to store its public keys in a storage system as the conventional and beneficial way to manage these keys. EX1014, 2:36-63. Bhattacharjee, ¶83 (citing EX1014, 2:35-63).

Because the certificate authority can be a third party, its storage system meets Google’s proposed construction of “*external storage system*” (“a storage system controlled by a third party”).⁶ EX1064, 12-13. Bhattacharjee, ¶¶84-86.

b. Alternatively, Orsini renders obvious [1A]

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.”

⁶ SFI has not currently proposed a construction of “*external storage system*.” See EX1065.

As discussed above (§VI.B.2.a), Orsini’s cryptographic engine receives a session master key, which is an “*external key*” under SFI’s proposed constructions. Sections VI.B.7.a.i-VI.B.7.a.ii below demonstrate how Orsini’s session master key is used to meet [1C]. Alternatively, sections VI.B.7.b.i-VI.B.7.b.iii below additionally discuss how other conventional encryption keys that are disclosed and/or rendered obvious by Orsini also meet [1C]. Bhattacharjee, ¶87.

If SFI argues that Orsini does not explicitly disclose receiving the session master key or any of the other conventional encryption keys discussed in §VI.B.7 below as an “*external key*” from an “*external storage system*,” POSAs had multiple reasons (discussed below) to implement Orsini to receive any of those keys as an “*external key*” from an “*external storage system*” under Google’s proposed construction and under the interpretation apparent from SFI’s infringement contentions, which identifies keys allegedly “stored centrally in Keystore, a repository built specifically for storing keys.” EX1009, 2. Bhattacharjee, ¶88.

First, as discussed, Orsini provides “***external*** session key management” and discloses “multiple cryptographic engines” having a “single point of...key management.” [0180], [0176]-[0179], [0293], [0381]-[0386], [0412]; Bhattacharjee, ¶89; EX1015, [0044]. POSAs would have recognized several benefits to implementing such a 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 discussed §VI.B.7.b *infra*) 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. Bhattacharjee, ¶89.

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 and storage demand on each cryptographic engine (by not requiring it to generate keys itself). Bhattacharjee, ¶90 (citing EX1020, 1:34-38; EX1021, [0032]; EX1055, [0006], [0009]; EX1067, [0004], [0016]). This obvious implementation 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. Bhattacharjee, ¶91 (citing EX1020, 1:34-38; EX1021, [0032]; EX1044, [0006], [0009]).

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, discussed *infra* §VI.B.7.b.ii). Bhattacharjee, ¶¶92-93 (citing EX1016, [0013]; EX1017, 1:65-2:2;

EX1018, 5:19-44; EX1019, [0034]; EX1022, [0004]-[0006], [0015]-[0017]; EX1023, 1:34-36; EX1020, 1:34-38; EX1021, [0032]; EX1055, [0006], [0009]).

This obvious implementation would have also beneficially reduced the risk of theft and/or sabotage of encryption keys by rogue insiders (*e.g.*, disgruntled employees) by preventing them from having direct access to the key-management system, which would be maintained in a secure location (separate from “*data chunks*” stored in systems for non-key data). Bhattacharjee, ¶¶94-95.

Second, receiving Orsini’s encryption keys from a centralized key-storage system built for generating and storing keys (separately from “*data chunks*”) would have been a simple substitution of one known element (Orsini’s storage system, if not external) for another (a conventional, centralized key-management system). *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 417-18 (2007); Bhattacharjee, ¶¶96-97; EX1022, [0003]-[0005] (POSA’s background knowledge of “keys...stored on a centralized server that is itself secured from access by attackers,” thereby keeping cryptographic keys secure and improving “security of the encrypted data”); EX1027, 3:27-35, 6:18-41.

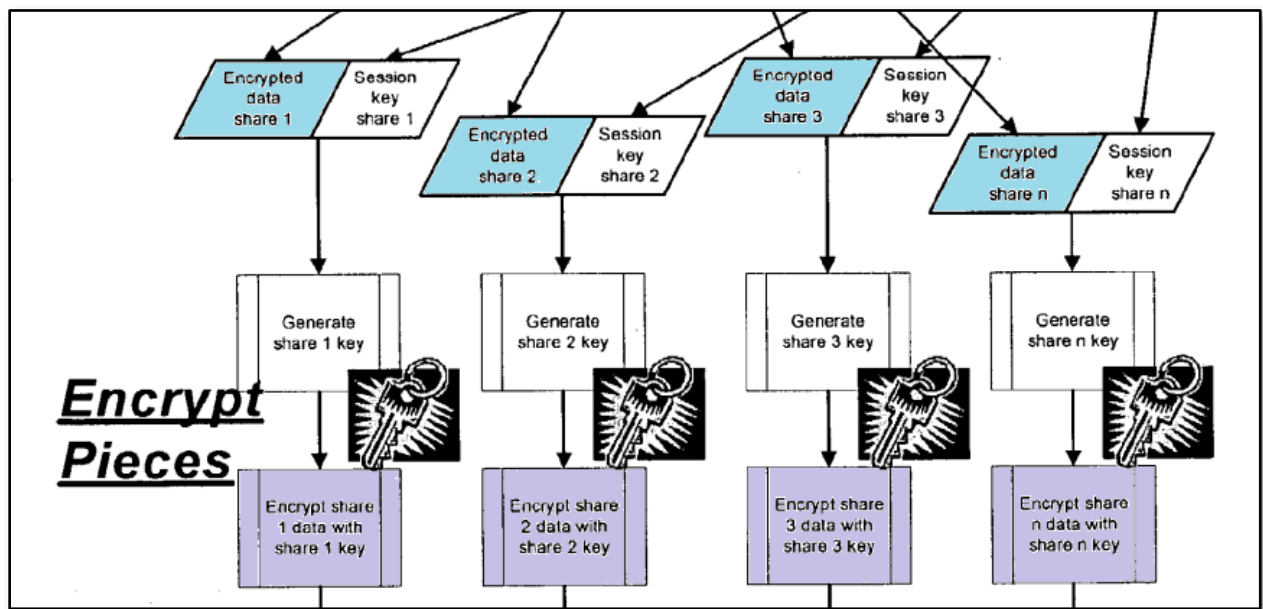
For additional security (*e.g.*, to better prevent the rogue-employee and sabotage problems discussed above), POSAs also had reason to implement the above-discussed key-management systems under the control of a third party (*e.g.*, in a secure, external facility under third-party control), which meets Google’s

construction of “*external storage system*” as “a storage system controlled by a third party.” Bhattacharjee, ¶98; EX1024, [0003]-[0010]; EX1025, 4:21-59, FIGS. 1-2; EX1064, 11-13.

POSAs would have reasonably expected success in each of the above implementations of Orsini’s encryption key-management system. Orsini’s “[e]ncryption key management...may be integrated into an existing infrastructure **or any other desired location.**” [0280]. Orsini also say keys may be stored in “a standard key store” or “custom key store.” [0392]-[0394]. It would have thus required only 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. Bhattacharjee, ¶99. The ’854 patent itself provides no details about, ascribes no benefits to, and identifies no challenges in implementing an external storage system for an external key. EX1001, 77:58-63, 78:66-79:23, 79:66-80:4; *Uber Techs., Inc. v. X One, Inc.*, 957 F.3d 1334, 1339 (Fed. Cir. 2020) (patent-at-issue’s “silen[ce] on how to” achieve claimed functionality suggests POSA “was more than capable of selecting between the known methods of accomplishing” it). Bhattacharjee, ¶100.

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

Orsini splits “the data into N **shares**,” where N is a plurality. [0280]-[0281], FIG. 21 (showing at least 4 shares). In Orsini, as illustrated in Figure 21 (color-annotated below), each **share** of the data set is encrypted with a distinct encryption key. [0281]. As discussed §§VI.B.4-VI.B.6 *infra*, the encrypted shares are “**data chunks**” as claimed. POSAs thus understood Orsini to be “*generating a plurality of data chunks based on the data set*” under SFI’s construction – by “creating a plurality of data chunks based on the data set” (EX1065, 2-3).⁷ Bhattacharjee, ¶101.



Orsini, FIG. 21 (partial; color-annotated)

⁷ Google has not proposed a construction of “*generating a plurality of data chunks based on the data set.*” EX1064.

Orsini also provides “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,” thereby reconstructing the data set. [0282]-[0284]. For example, 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.” [0283]. Thus, Orsini’s “*data set can be reconstructed using at least a minimum number of the plurality of data chunks*” under both parties’ proposed constructions, because Orsini’s “subset of the shares,” once encrypted, are a “predefined number [*e.g.*, three] of the lowest amount of the plurality of data chunks” (Google; EX1064, 13)) and “a threshold number of chunks less than or equal to the total number of data chunks” (SFI; EX1065, 9) from which the data set can be reconstructed after decryption. Bhattacharjee, ¶102.

**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,**

As illustrated 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” (Google) and “dividing and placing” (SFI) “*the data set into a plurality of shares.*” EX1064, 2-5; EX1065, 2; *see also* Orsini, [0117], [0281]-[0282], [0294]-[0296]. Bhattacharjee, ¶103. As also illustrated below, each **share** comprises a “portion[] of the original data” ([0298]) and thus “*each of the shares comprises less than all of the data set,*” as claimed.

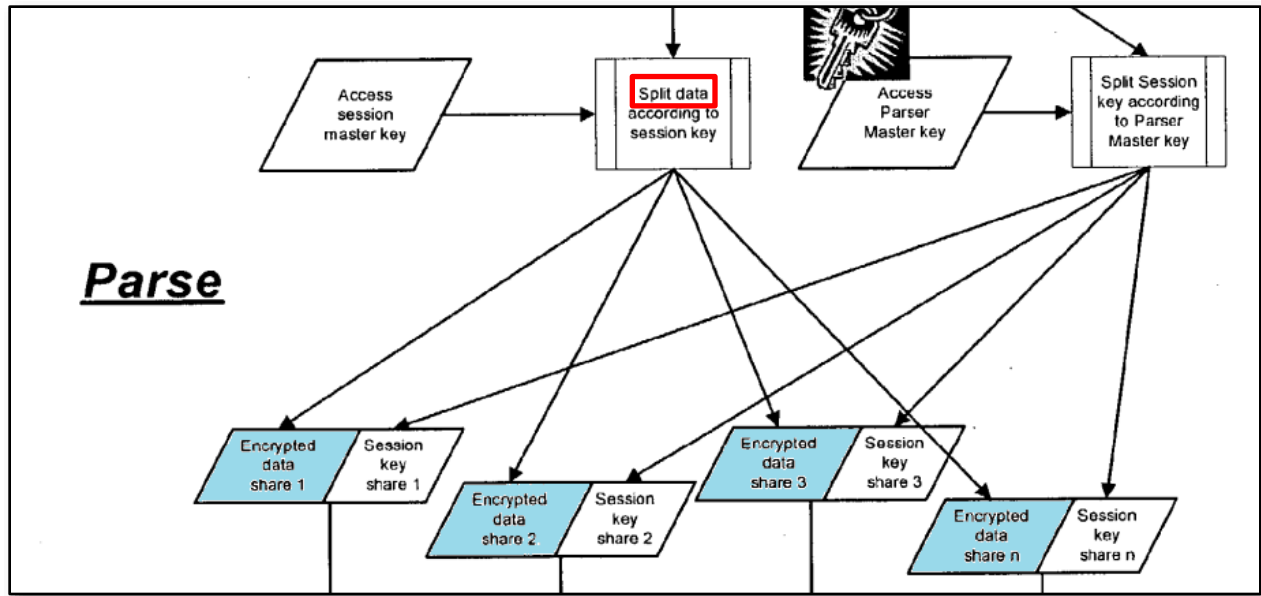


FIG. 21 (partial; color-annotations added)

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

As discussed below (§VI.B.6 [1B3]), Orsini’s parser program encrypts each **share** with a distinct **share encryption key**, and so is “*accessing a plurality of distinct encryption keys*,” as claimed. [0298]-[0299]; see also [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**”); Bhattacharjee, ¶104.

If [1B2]’s “*accessing*” requires the “*distinct encryption key*” to pre-exist, rather than be generated on-the-fly, that would have been obvious for the same reasons discussed *supra* §VI.B.2.b that it would have been obvious to generate, distribute, and store encryption keys from a centralized key-management system. Bhattacharjee, ¶105.

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

As Figure 21 illustrates (color-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 **share encryption key**. [0298]-[0299]; see also [0346]-[0352] ("multiple *distinct* secret key values: K1, K2, K3, Kn, K5" used to encrypt respective shares), [0395]-[0396]; Bhattacharjee, ¶106.

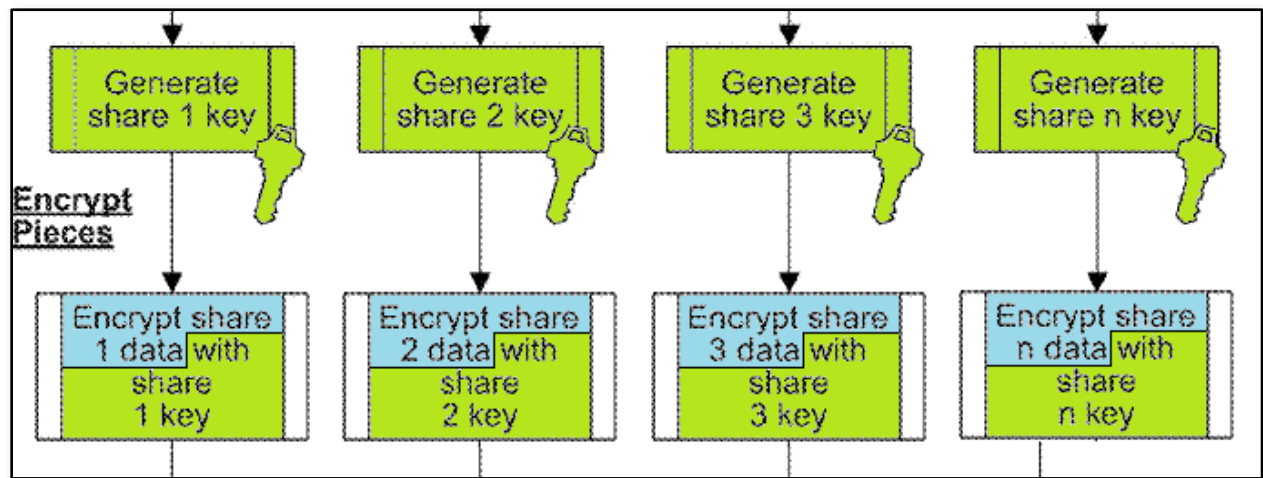


FIG. 21 (partial; color-annotations added)

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

Orsini discloses "performing an encryption operation based on the external key to further secure the plurality of data chunks," as claimed. Alternatively, implementing Orsini such that its encryption method included that step would have been obvious in view of Orsini's disclosure. Bhattacharjee, ¶107.

a. Orsini discloses [1C]

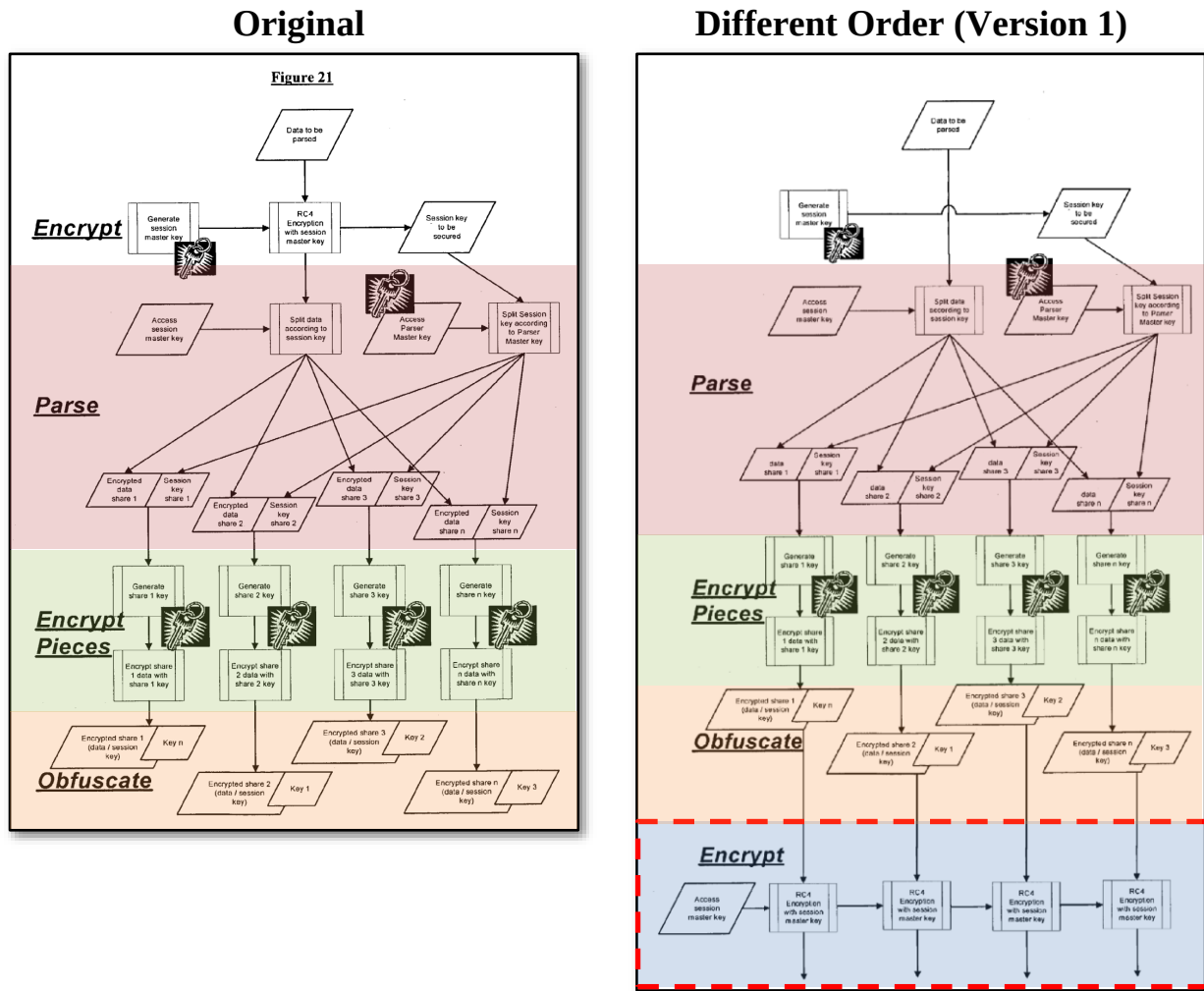
Orsini discloses [1C] in three independent ways.

i. Performing encryption steps in different order.

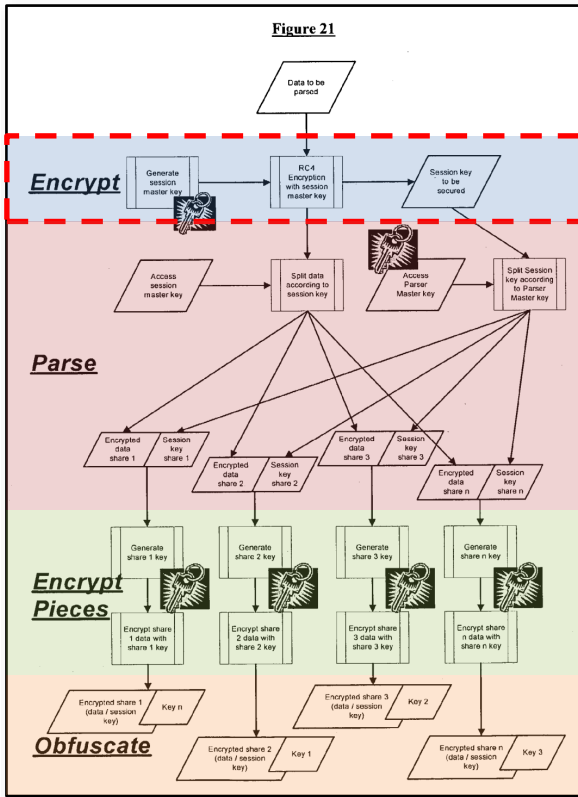
First, in discussing Figure 21’s encryption method, Orsini teaches that “[i]t is readily apparent” to POSAs “that certain steps of the method[] described herein may be performed **in different order.**” [0301]; *see also* [0279] (noting “[t]he parsing process” is “designed in a modular tiered fashion,” which suggests and reinforces the ability to change the order of encryption processing); Bhattacharjee, ¶¶108-09. Orsini thus discloses embodiments in which “certain steps” in 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. Bhattacharjee, ¶109.

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 (discussed *supra* §VI.A) because it was well-known to reorder whether an encryption step is performed before and/or after splitting. Bhattacharjee, ¶110-11 (corroborating with EX1028, [0029]; EX1053, [0026]; EX1045, FIGs. 11-13, [0109]-[0113]). Thus, one different-order embodiment includes the session master key (an exemplary “*external key*”) further encrypting 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

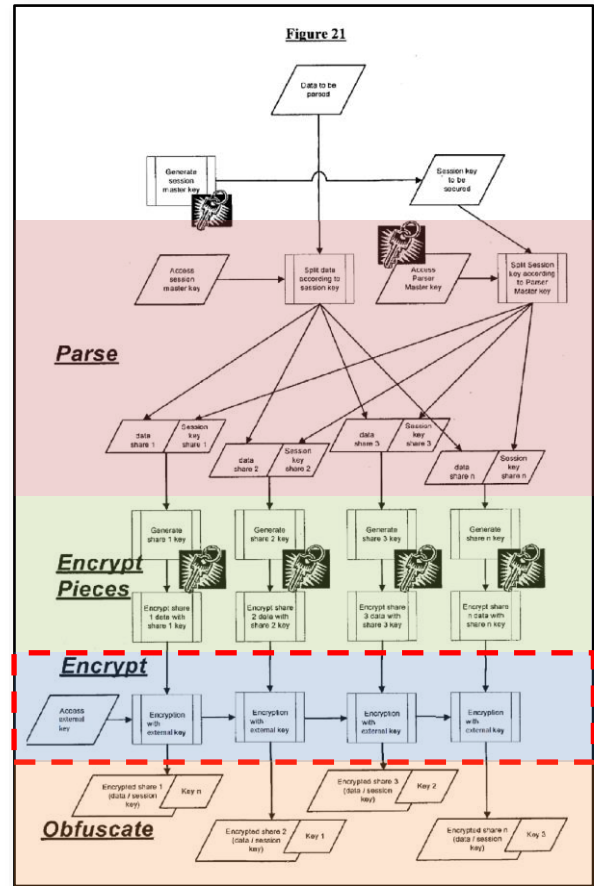
before either of those steps. In this embodiment, “*data chunks*” are “*further secure[d]*” as claimed, because they undergo a second encryption operation using the session master key, which meets the “*external key*” as explained *supra* §VI.B.2 ([1A]). Two exemplary implementations of that embodiment are conceptually illustrated below (right):



Original



Different Order (Version 2)



Bhattacharjee, ¶¶112-13.

ii. Repeating encryption steps.

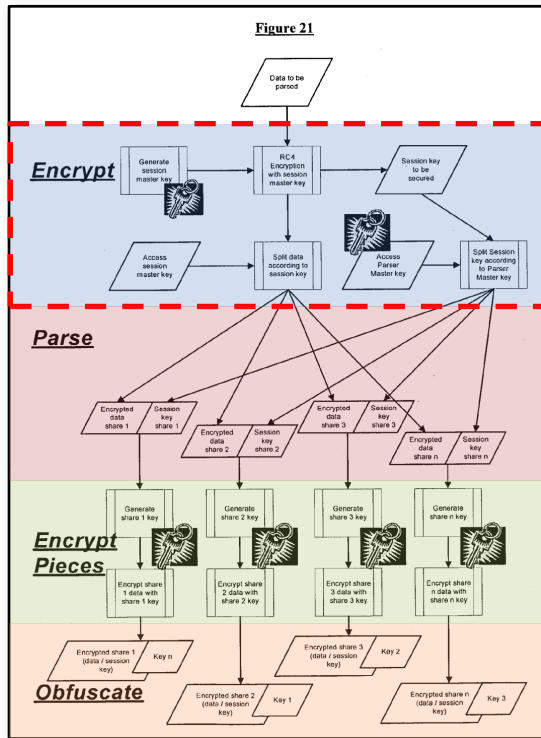
Second, Orsini also teaches that Figure 21's steps can be "**repeated multiple times.**" [0301]. Indeed, Orsini has dependent claims that recite encrypting parsed data portions **a second time** using "a different encryption algorithm" than the algorithm used to encrypt the data portions the first time. See, e.g., Claims 12 (claim 10's encryption "step b)" which encrypts portions of data "is repeated using a different encryption algorithm"), 29 (similar), 39 (similar); Bhattacharjee, ¶114.

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 (thus performing a second encryption after the encryption that follows splitting the data set). Bhattacherjee, ¶115.

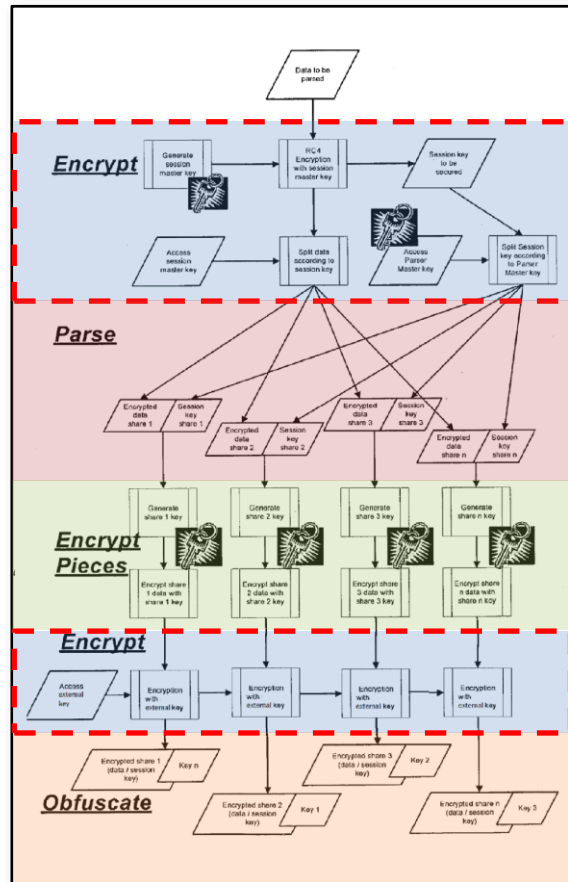
One different algorithm performed after encrypting the “*shares*” (thus, after generating “*data chunks*”) that would have been obvious to POSAs includes using a single encryption key to further encrypt all data chunks with that key, similar to using the session master key to encrypt the entire data set originally before splitting it. In other words, Figure 21’s method would have five steps, not four: (1) encrypt the original data set with a session master key, (2) split the data into shares, (3) encrypt the data shares with distinct share keys, (4) **encrypt each encrypted share (the “*data chunk*”) with a common key**, and (5) obfuscate the encrypted shares. The common encryption key used to further encrypt the already-encrypted shares (the “*data chunks*”) would have been, or had reason to be, an “*external key*” for the reasons discussed *supra* §VI.B.2 ([1A]). Bhattacherjee, ¶116.

Two exemplary implementations of that embodiment are conceptually illustrated below (right):

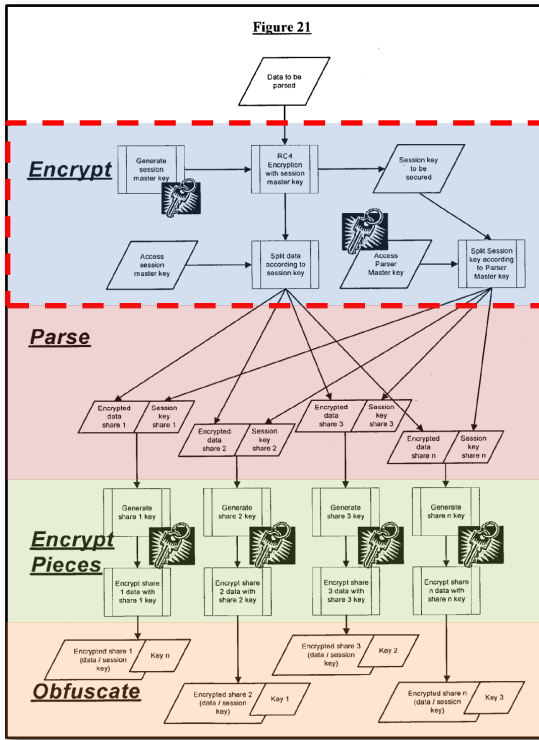
Original



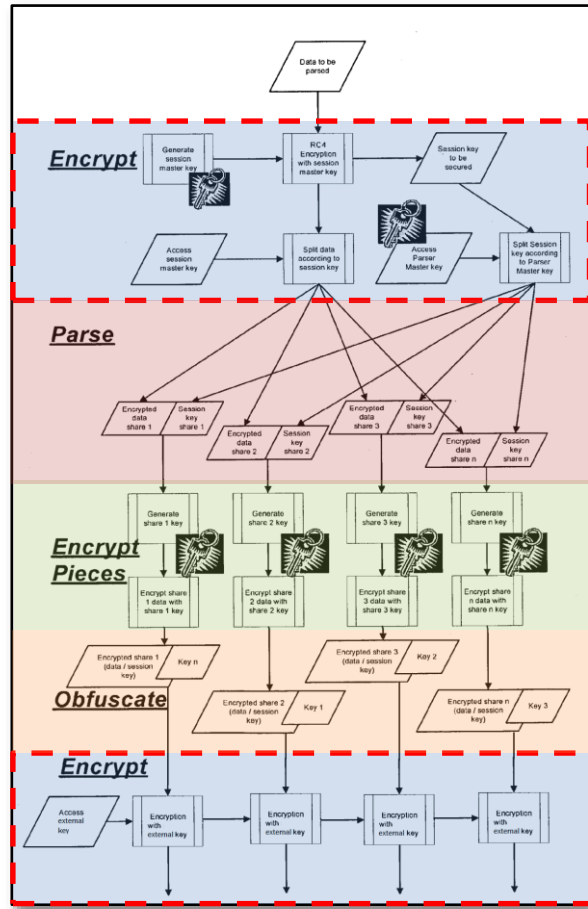
Two "Encrypt" Phases (Version 1)



Original



Two “Encrypt” Phases (Version 2)



Bhattacharjee, ¶117.

iii. Encrypting with a public key.

Third, as discussed *supra* §VI.B.2.a.ii ([1A]), Orsini’s public encryption keys are additional “external keys” under both parties’ proposed constructions. Orsini uses the public encryption key(s) to further encrypt already-encrypted “data chunks.” [0097], [0107] (“cryptographic engine” can perform “public-key encryption”). “For example, public-key encryption may be used to **further secure the data** at the data storage facilities.” Orsini, [0119], [0095] (disclosing “incoming”

“communication” to the depository “may advantageously be encrypted with public key encryption”). That encryption operation includes further encrypting each encrypted share (“*data chunk*”) with a public key before transmitting the further-encrypted chunks to the data storage facilities, thereby “*further secur[ing] the plurality of data chunks*,” as claimed. Bhattacharjee, ¶118.

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.” [0172]. As discussed above (§VI.B.7.a.i), an exemplary session key that Orsini uses to encrypt data is the session master key. Under the scope of the claims that SFI attributes to [1C] in related district-court litigation, encrypting [1B3]’s “*distinct encryption keys*” with the “*external key*” meets [1C]. EX1009, 8. If that were correct, then encrypting Orsini’s session master key with a public key (an “*external key*”) would also meet [1C]. Bhattacharjee, ¶119.

b. Alternatively, [1C] is obvious over Orsini

Alternatively, POSAs had multiple reasons to implement Orsini to perform [1C] in view of POSAs’ background knowledge.

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

As discussed *supra* §VI.B.7.a.i, Orsini teaches “perform[ing]” “certain steps of the method” “*in different order*” or “*repeated multiple times*.” [0301]. POSAs would have been motivated to reorder and/or repeat certain specific steps by

encrypting already-encrypted data chunks with the same encryption key, and would have understood that doing so would provide additional security and efficiency benefits. Bhattacharjee, ¶¶120-22; *see supra* §VI.B.a (demonstrating Orsini discloses or alternatively renders obvious receiving its encryption keys from an “external storage system”); *infra* §§VI.B.7.b.ii-iii (discussing benefits of further encrypting data chunks), §VI.B.7.b.iv (discussing known benefits of key-encryption-keys (“KEKs”), which SFI contends meets the limitation, *see* EX1009, 8).

Re-encrypting data chunks with a common external encryption key was also a known solution to the known problem of how to further secure data using encryption, and thus was obvious. *Intel Corp. v. PACT XPP Schweiz AG*, 61 F.4th 1373, 1380 (Fed. Cir. 2023) (“[I]f there’s a known technique to address a known problem using prior art elements according to their established functions, then there is a motivation to combine.”) (cleaned up). Bhattacharjee, ¶123 (citing, *e.g.*, EX1028, [0090]; EX1053, [0026]).

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

Orsini discloses 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, [0412]. A conventional implementation of such

embodiments would have been to encrypt data chunks with a group-level encryption key, thereby restricting user access to chunks based on membership in an authorized group(s). *KSR*, 550 U.S. at 417 (obvious to use known technique to improve similar devices); Bhattacharjee, ¶124 (explaining that group-level encryption was well-known and conventional); see EX1018, 5:19-35 (storing group-level keys in a “data key data store 45” that can “be set up centrally at a remote location” to allow “it to be isolated” and have “stringent security precautions”); EX1029, 6:55-8:57 (encrypting different portions of a document with different keys to allow multiple levels of access); EX1033, 8:31-41 (encrypting a file key with a “group key” so that only an approved “member of a group” can decrypt the file); EX1034, [0033] (“associat[ing] encryption keys with each group of users” and encrypting “portions of documents...with the corresponding group’s encryption key”).

POSAs had reason for the group-level encryption keys to be received from an external storage system for the same reasons discussed above (§VI.B.2, [1A]) that POSAs had reason to receive any other encryption keys from an external storage system. Bhattacharjee, ¶125.

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

POSAs also had reason to add additional security to Orsini’s encrypted shares (“*data chunks*”) by cryptographically reordering/shuffling the data *within* each “*data chunk*,” which was a conventional cryptographic operation to secure data.

EX1030, [0002]. Bhattacharjee, ¶¶126-27. 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.” [0281]. POSAs understood that the benefits from shuffling data would also accrue if the shuffling were performed after encrypting each share with a respective encryption key (that is, shuffling the data within each “*data chunk*”). And, as noted above (§VI.B.7.a.i), Orsini explicitly teaches performing encryption steps in different order. Bhattacharjee, ¶126.

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). Bhattacharjee, ¶127.

In a conventional technique, the reordering pattern for portions (*e.g.*, bits/bytes) of data would be defined using an encryption key. Bhattacharjee, ¶¶128-29; EX1031, FIGS. 1-4, [0002]-[0003], [0016]-[0017], [0021]-[0023]; EX1032, FIG. 5, 1:4-36, 4:3-26, 6:23-7:16; EX1040, [0030]; EX1041, [0024].

Thus, POSAs had reason to further secure the encrypted shares (“*data chunks*”) based on one of the external keys discussed above (§VI.B.2), as claimed, by using the received external key to define the reordering pattern within each “*data chunk*.” Bhattacharjee, ¶130.

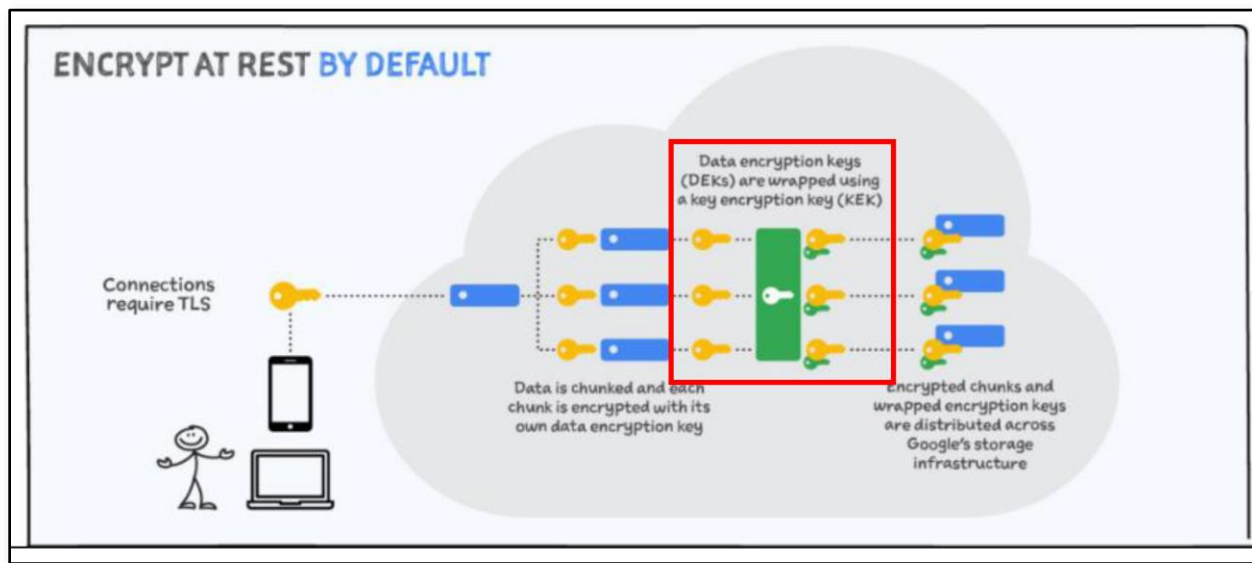
iv. POSAs had reason to encrypt [1B3]’s distinct encryption keys with an externally-stored key-encryption key

As discussed above (§VI.B.7.a.iii, [1C]), Orsini discloses (or alternatively renders obvious) encrypting session keys with a public-key encryption key. [0172]. Thus, Orsini discloses embodiments that encrypt encryption keys with other encryption keys, the latter of which are known in the art as key-encrypting keys (“KEKs”). Bhattacharjee, ¶131 (citing EX1035, EX1036).

As discussed below (§VI.B.8, [1D]), Orsini stores [1B3]’s distinct encryption keys (with “*data chunks*”) on storage systems. However, POSAs knew, in general, “cryptographic keys should never occur in clear form outside the cryptographic device.” EX1035, 12:15-23, 10:42-62; *see also* EX1036, 1:66-2:6 (when “encryption keys are included with the encrypted message” “[s]ecurity” comes from “encrypt[ing] the encryption keys themselves” with “a key encryption key”). Thus, to avoid a potential security risk in storing [1B3]’s encryption keys in clear (unencrypted) form on storage devices (especially those accessible from the public Internet), POSAs had reason to encrypt [1B3]’s keys with a KEK. Bhattacharjee, ¶132 (citing EX1035 at 8:15-22, which explains that there are “two categories of cipher keys”: “operational keys (KO) and key encrypting keys (KEK)”).

As shown below, under the scope of the claims that SFI attributes to [1C] in its district-court complaint, encrypting [1B3]’s keys with a KEK meets [1C]. *See*

EX1009, 8 (arguing “DEKs are encrypted with (wrapped by) a key encryption key (KEK)”). POSAs had reason to receive the KEK from an “*external storage system*” for the same reasons POSAs had to receive Orsini’s session master key from an external storage system, as discussed above (§VI.B.2.b, [1A]). Bhattacharjee, ¶133.



* * *

POSAs would have had a reasonable expectation of success in any of the conventional encryption operations discussed above in this §VI.B.7.b (*e.g.*, repeating/reordering individual encryption steps, reordering data within chunks, group-level encryption keys and/or KEKs), because those were conventional, well-known encryption operations, and because Orsini discloses that “[e]ncryption of data may be performed prior to or after the first parsing, splitting or separating of the data” and that “the encrypting step may be repeated for one or more portions of the

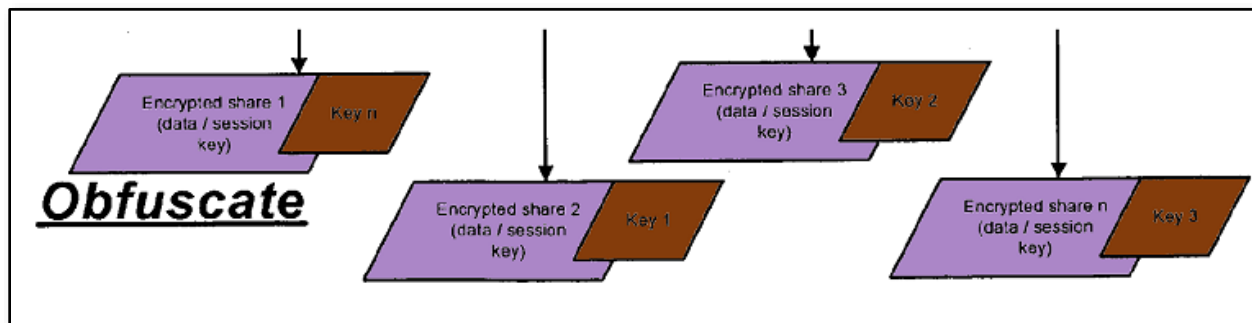
data” ([0013]), and further discloses that “a variety of combination of [encryption method] steps may be desired” ([0230]). Bhattacharjee, ¶134.

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.

With respect to data stored with [1B3]’s distinct encryption keys, Orsini’s disclosure is substantively identical to the ’854 patent’s disclosure. For example, SFI cites EX1001, 31:39-45, 48:33-43, and 74:62-66 as intrinsic evidence that allegedly supports its construction of the “*data indicative*” term. See EX1065, 4. EX1001, 31:39-45 and 48:33-43 are identically disclosed in Orsini at [0162] and [0247], respectively. EX1001, 74:62-66 uses 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” ([0134]) “partition[ing] the data” “based on...random” values ([0280]-[0281]). Thus, Orsini discloses [1D] to the same extent the ’854 patent does. Bhattacharjee, ¶¶136-37.

As Figure 21 illustrates (partially shown and color-annotated below), Orsini stores **encrypted shares** (“*data chunks*,” §VI.B.3) with a share encryption key (“*distinct encryption keys*,” §VI.B.5). The share encryption key itself is “*data indicative*” of that key under SFI’s construction, which includes an encryption key itself. EX1065, 3-4 (data that “makes known” or “shows” the key); see EX1009, 9

(alleging [1D] is met by “storing DEKs”—the keys alleged to meet [1B]—“with the data chunks”).



The share encryption keys that encrypt shares are stored “in different locations from the encrypted data portions or shares” (e.g., “Share 1 gets Key 4” etc.). [0299]. These different locations store the **encrypted share** (“*data chunk*”) and **share encryption key** (“*distinct encryption key*”) “on one or more data depositories or *storage devices*” (“*plurality of different storage devices*”), as claimed. [0379]; see [0018], [0111]-[0112], [0154]-[0155], FIGs. 7, 21, Claims 10, 13-16; Bhattacharjee, ¶137.

POSAs also had reason to store with each encrypted share “*data indicative of*” a share encryption key under Google’s construction, which requires “metadata about” the key. EX1064, 5-7. As discussed above (§VI.A), Orsini supported multiple prior-art encryption algorithms, like AES. [0378]. POSAs understood that AES (and other conventional encryption algorithms) supported multiple so-called encryption modes. Bhattacharjee, ¶138 (citing EX1039, 2:58-3:11). Thus, POSAs had reason to store each encrypted share with data indicating the encryption mode

associated with a share encryption key, which is “metadata about” the encryption key, and thus meets Google’s constructions. EX1064, 5-6. Bhattacharjee, ¶138.

POSAs also understood that it was conventional to support variable-sized encryption keys, to provide additional security. Bhattacharjee, ¶139 (citing EX1039, 2:58-67; EX1066, [0009]; EX1056, 3:8-24). When using those algorithms, POSAs had reason to store each share with metadata indicating the length of the share encryption key stored with the share, which also meets Google’s constructions. EX1064, 5-6. Bhattacharjee, ¶139.

Thus, Orsini discloses and renders obvious [1D] under both parties’ proposed constructions. Bhattacharjee, ¶140.

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. 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**,” which POSAs understood duplicates each chunk and stores each chunk’s copies on more than one storage device. [0388]; Bhattacharjee, ¶141.

As discussed above (§§VI.B.2-VI.B.8), Orsini performs limitations [1A]-[1D] in the order in which claim 1 recites them; and, further, doing so would have been a

conventional design choice. Orsini thus meets Google’s construction requiring claim 1’s steps to be performed in order. EX1064, 16. Bhattacharjee, ¶142.

C. Claim 2: [1D]’s storing...comprises storing each of the plurality of data chunks with at least one of the distinct encryption keys.

As discussed above (§VI.B.8, [1D]), each of Orsini’s encrypted shares (“*data chunks*,” §VI.B.3 [1B]) is stored with one of the share encryption keys (“*distinct encryption keys*,” §VI.B.6 [1B3])—specifically, a key that was used to encrypt different one of the shares. FIG. 21, [0299]; Bhattacharjee, ¶143; EX1009, 10-11 (alleging claim 2 is met by “storing a DEK with each data chunk”).

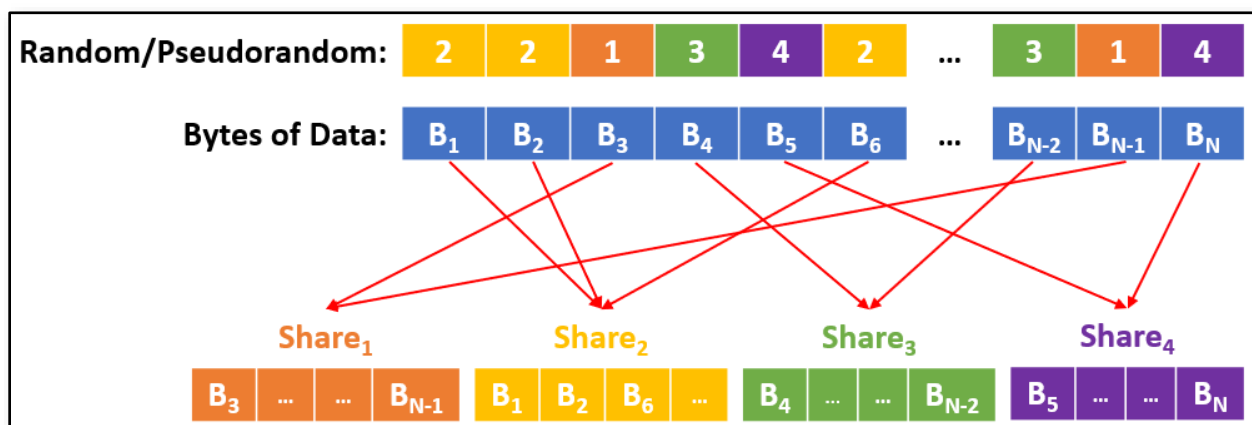
D. Claim 3: [1D]’s storing...comprises including with a particular data chunk of the plurality of data chunks data indicative of an encryption key that was used to encrypt a different data chunk of the plurality of data chunks.

Orsini meets claim 3 in the manner discussed *supra* §VI.B.8 ([1D]) and §VI.C (claim 2). FIG. 21, [0298]-[0299], [0379]-[0380], [0395]-[0396]. Bhattacharjee, ¶¶144-45.

E. Claim 4: [1B1]’s distributing...comprises using a random technique or pseudorandom technique.

As discussed above §VI.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. [0087], [0280], [0393]-[0394]. Orsini discloses several random/pseudorandom techniques to perform that step. Bhattacharjee, ¶146.

For example, Orsini discloses a “cryptographic split process” that distributes the data set into, *e.g.*, “4 shares” based on a “**random** distribution” of numbers “corresponding to the four shares,” which is using a random technique in distributing the data set into shares, as claimed. [0282]; *see also* [0105], [0116], [0124], [0281], [0284], Claim 54; Bhattacharjee, ¶¶147, 149 That technique is illustrated below (Bhattacharjee, ¶149):



Additionally, Orsini discloses that data can be “parsed into ‘n’ segments, or shares, according to the value of K5,” which is a “secret key value[.]” [0346]-[0348]. This “results in ‘n’ **pseudorandom** shares of the original encrypted data” ([0348]), and thus is using a pseudorandom technique, as claimed. Bhattacharjee, ¶148.

F. Claim 5: [1B1]’s distributing...comprises using a deterministic technique or pseudorandom technique.

Orsini meets claim 5 in the same manner as claim 4 using a pseudorandom technique, as discussed above (§VI.E). Bhattacharjee, ¶150.

Alternatively, Orsini’s “parsing” step distributes the data set into shares using a deterministic technique, by using “*predetermined*” (“same”) sized portions of the data set. [0116], [0281]; Bhattacharjee, ¶151.

G. Claim 6: ...each share...comprises a substantially random distribution of a subset of the data set.

Orsini meets claim 6 because the random/pseudorandom techniques discussed for claim 4 (*supra* §VI.E) create a substantially random distribution of a subset of the data set in each share. Bhattacharjee, ¶152.

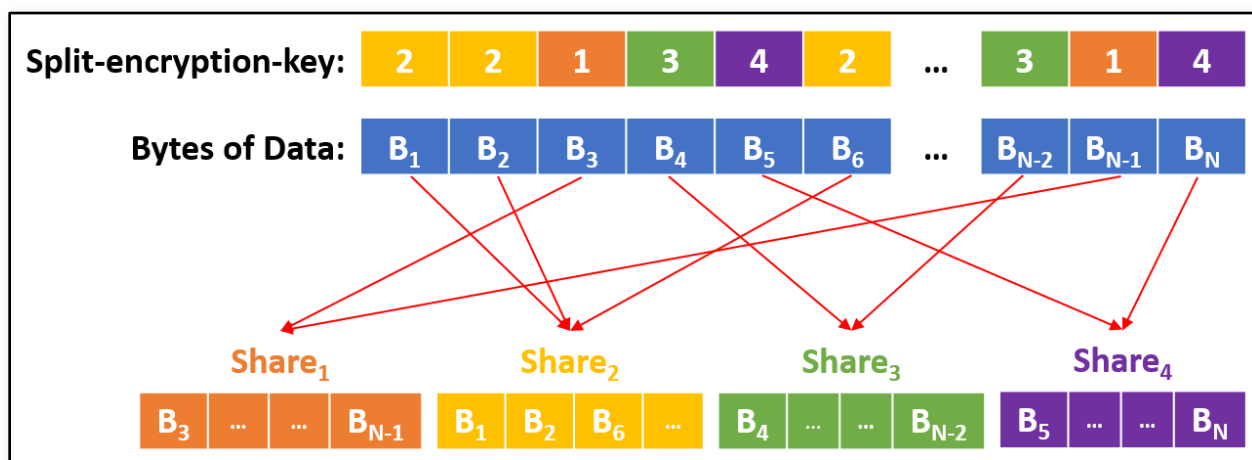
H. Claim 7: [1C]’s encryption operation...comprises encrypting primary data with the external key.

Orsini discloses “us[ing]” the session master key (“*external key*,” *see supra* §VI.B.2.a.i, §VI.B.2.b) “to encrypt the data prior to the parsing operation.” [0393]-[0394]. Claim 7 recites “*primary data*” without antecedent basis, and the ’854 patent’s specification does not explain what “*primary data*” means. If “*primary data*” is the “*data set*” before it’s distributed into shares (§VI.B.4), Orsini meets “*encrypting primary data with the external key*” because, as discussed (§VI.B.7.a.ii, §VI.B.7.b.i), Orsini discloses or alternatively renders obvious repeating an encryption step with an “*external key*” both before splitting Orsini’s data set and after encrypting its individual shares. Bhattacharjee, ¶153.

If SFI argues claim 7 is met by encrypting [1B3]’s keys (Orsini’s share encryption keys) with a key-encryption-key (KEK), POSAs had reason to do so for the reasons discussed (§VI.B.7.b.iv). Bhattacharjee, ¶154.

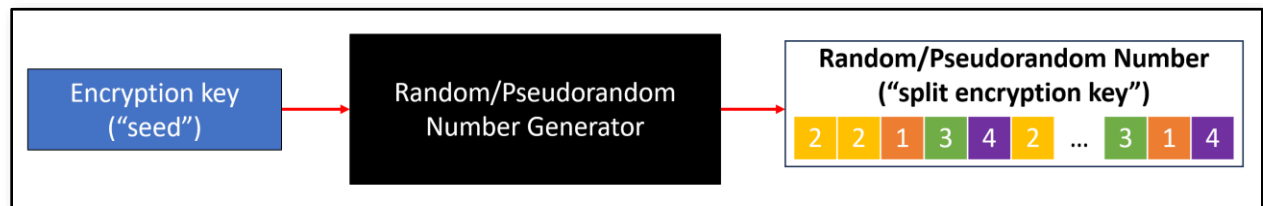
I. Claim 8: [1B1]’s distributing...comprises: generating a split encryption key based on the external key; and distributing the data set based on the split encryption key.

As discussed (§VI.E, claim 4), Orsini discloses distributing the data set into 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.” [0282]. As illustrated below, that number sequence is a “*split encryption key*” because it is the key used in the algorithm for splitting data into shares. Bhattacharjee, ¶155. See EX1001, 78:11-28.



Orsini says the random/pseudorandom number (“*split encryption key*”) is itself created using a key ([0282]), which is consistent with a conventional

random/pseudorandom number generator (illustrated below) that inputs an encryption key as a so-called “seed.” Bhattacharjee, ¶¶156-57 (citing EX1040, [0030]; EX1041, [0024], [0026]-[0027]).



As discussed (§VI.A), Orsini distributes data based on the session master key (“*external key*”—*supra* §VI.B.2.a.i, §VI.B.2.b). [0296], FIG. 21. Thus, in embodiments using a random/pseudorandom technique to distribute data, Orsini’s session master key is the key used to generate the random/pseudorandom number (a “*split encryption key*”). Alternatively, POSAs had reason to generate that key using the session master key because it would have been a simple substitution of one technique for generating a random/pseudorandom number for another, to obtain the predictable result of generating random/pseudorandom numbers based on an encryption key. Bhattacharjee, ¶¶158-59.

Thus, Orsini discloses, or alternatively renders obvious, using a “*split encryption key*” that is based on the session master key (implemented as an “*external key*”), and distributing data into shares based on that split encryption key, as claimed. Bhattacharjee, ¶160.

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

As discussed, Orsini discloses and renders obvious (1) claim 8 (§VI.I) and (2) [1C] (§VI.B.7) by encrypting all shares with the same “*external key*” (*e.g.*, the session master key) as claimed. Bhattacharjee, ¶161.

For example, Orsini discloses (§VI.B.7.a.i) and renders obvious re-ordering Figure 21 to encrypt “*data chunks*” (rather than the original data set) with the session master key (§VI.B.7.b.iii) or a group-level key (§VI.B.7.b.iii). Alternatively, Orsini also discloses (§VI.B.7.a.ii) and renders obvious (§VI.B.7.b.i) repeating the encrypting of data with the session master key. Each of those embodiments encrypts the shares with the (“*external*”) session master key and/or group key, as claimed. See [0119] (“further secur[ing] data chunks” with a public key). Bhattacharjee, ¶¶162-63.

Finally, if SFI argues encrypting [1B2]’s encryption keys with a key-encryption key (KEK) meets claim 9, it would have been obvious to do so using a KEK received from external storage, for the reasons discussed above (§§VI.B.2.b, VI.B.7.b.iv). Bhattacharjee, ¶164.

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.

Orsini meets claim 10 for the reasons discussed above for [1B1] (§VI.B.8) and claim 9 (§VI.J), because “[t]o restore the original data format, the steps are reversed.” [0300]. Bhattacharjee, ¶165.

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

A. Foster (EX1006)⁸

Foster discloses methods for securing digital content using encryption techniques that “attach[] a title [encryption] key to encrypted content for synchronized transmission to, or storage by, a recipient.” Abstract, [0002]. Foster’s digital content includes “*any* data” (*e.g.*, digital movie). [0037]; Bhattacharjee, ¶¶165-66.

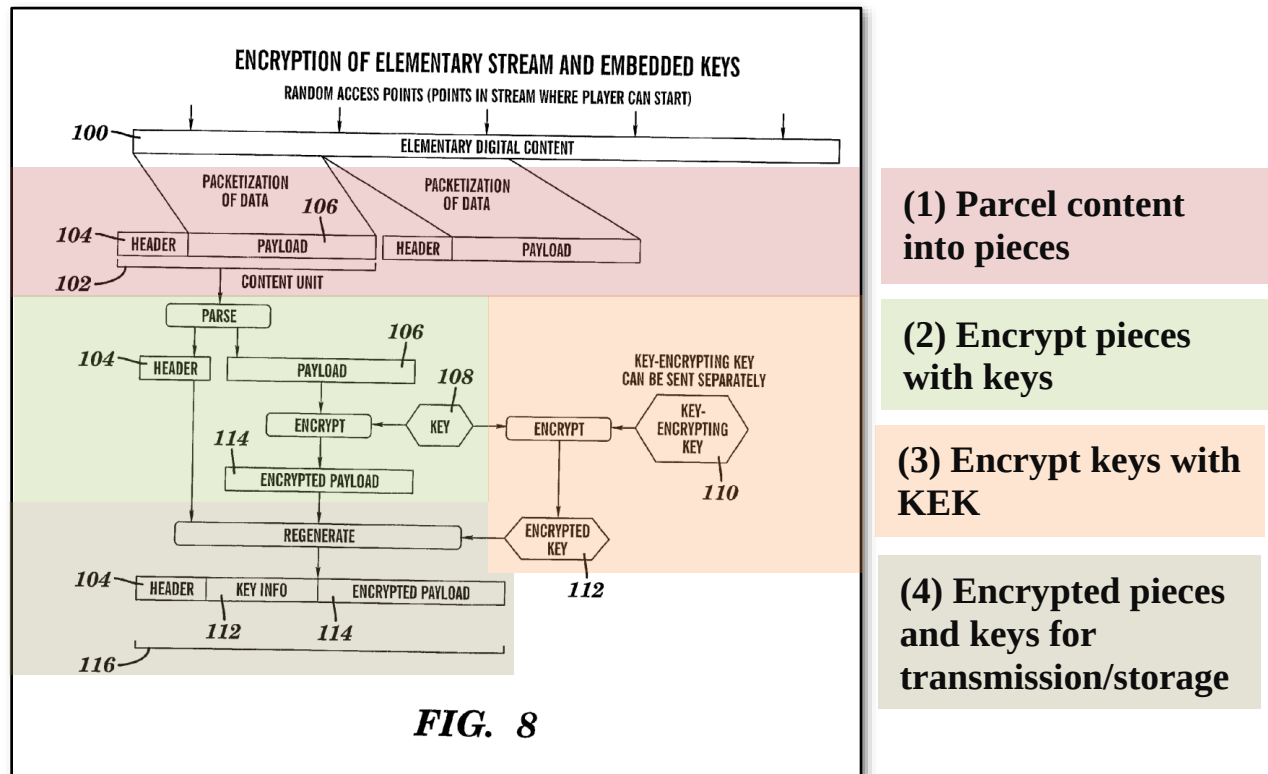
Foster teaches that prior-art encryption methods used “multiple keys” to “provide[] enhanced security.” [0004]. Those techniques, however, typically stored encryption keys separately from the encrypted content portions, and therefore required subsequent “synchronization” of each encrypted portion with its corresponding encryption key, which also created challenges in “provid[ing] random access” to portions of content. [0006]; Bhattacharjee, ¶168.

To address these deficiencies, Foster’s encryption methods involve (1) parceling (splitting) content 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

⁸ Unless otherwise noted, Section VII.A’s citations are to Foster.

packets, and (5) transmitting the encrypted content packets to receivers (*e.g.*, devices in a consumer's home). [0040]-[0041], [0064], FIG. 8; Bhattacharjee, ¶169.

An embodiment of Foster's method is illustrated in Figure 8 (annotated below):



First, “content stream 100” “is parceled into content units 102.” [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”). [0064], FIG. 8.

Second, each payload portion is “encrypted with [a] title key 108” to “yield encrypted content packet 114.” [0064]. Instead of one title key for all portions, “each content packet could be encrypted with a *different* title key.” [0065].

Third, each title key is encrypted with the same “key encrypting key 110” (a KEK, *see* §VI.B.7.b.iv *supra*). Each “encrypted title key 112” is then “attached to the encrypted content as a header extension.” [0048], [0064], [0066], FIGS. 8-10.

Fourth (and finally), the encrypted title key, encrypted content packet, and header are combined as a “processed content unit 116,” which can be “stored by a recipient” (*e.g.*, in a consumer’s set-top box). [0040]-[0041], [0064]; Bhattacharjee, ¶¶170-74.

Foster’s Figure 16 (color-annotated below) depicts an exemplary system employing its encryption technique. [0030], [0075]. Foster’s KEK discussed above is stored in “[d]atabase 426” (beige). [0046]; *infra* §VII.B.2. The system also includes a “source 428” (green), which can be a “content owner...such as a movie studio.” [0038], [0042], [0078]. It also includes “recipient 430” (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.” [0039]-[0041], [0078]; Bhattacharjee, ¶¶175-76.

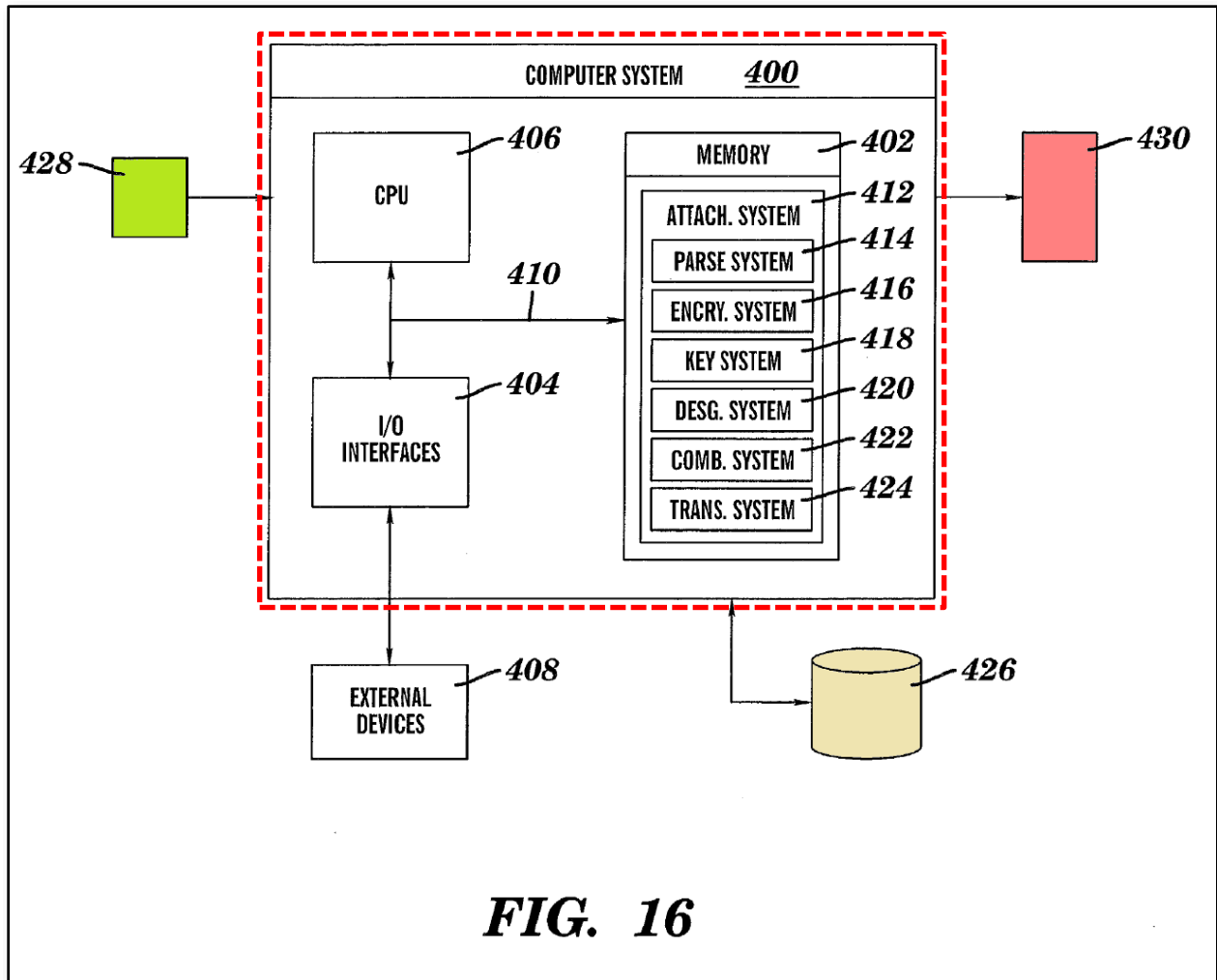


Figure 16's system includes "computer system 400" (with a computer in the red-dashed box) which is "any system capable of processing content units of an elementary content stream." [0078]. Because system 400 is interposed between source 428 and recipient 430, POSAs understood Foster to be disclosing that system 400 can perform functions of the "Content Service Provider" that is acting as a "pipeline" to deliver data from content owner 428 to recipients 430 (*e.g.*, residential

customers that rent digital movies from cable (content service) providers).
Bhattacharjee, ¶177.

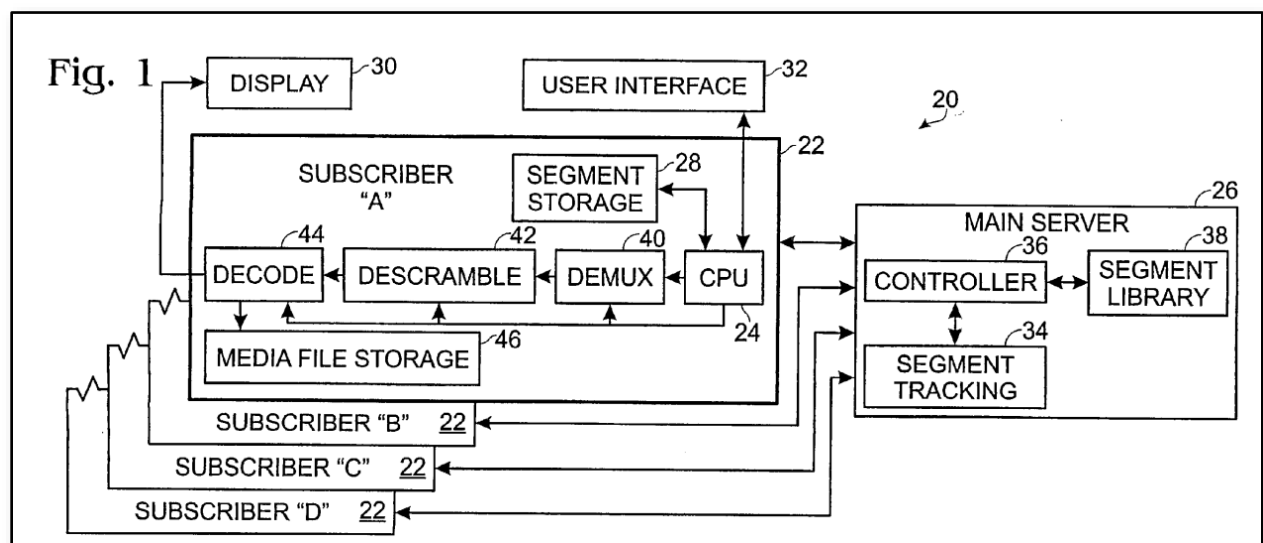
B. Hayhurst (EX1007)

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 data “delivery-on-demand” “require[d] an extensive amount of bandwidth.” Hayhurst, [0004]. To address that problem and provide other advantages, Hayhurst’s media files are segmented and “distributed among [a] network of subscriber units,” so individual subscribers can retrieve a media file’s segments from storage devices maintained by other subscribers. Hayhurst, [0006]. Like Foster’s, Hayhurst’s “segments are each encrypted...to prevent unauthorized viewing of individual segments.” Hayhurst, [0007]; Bhattacharjee, ¶178.

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]). Each segment is also duplicated and 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]. Bhattacharjee, ¶179.

Hayhurst’s Figure 1 (below) illustrates several subscribers units (subscribers A-D) “in communication with a main server 26” (*e.g.*, at the content provider) and each unit is “a conventional set-top box” (just like “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, [0017]-[0018]; Foster, [0040]; Bhattacharjee, ¶180.



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, [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.” Hayhurst, [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 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]. Bhattacharjee, ¶181.

C. Foster-Hayhurst

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. Bhattacharjee, ¶182.

POSAs would have recognized several benefits in storing Foster’s encrypted content units distributed on different storage devices as Hayhurst teaches. Foster’s encryption method can be implemented in a computer system at a content service provider. *See* §VII.A *supra*. Similarly, Hayhurst teaches benefits 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, [0005]), “provides a high level of security and a deterrence to piracy” (Hayhurst,

[0008]), and supports redundancy in segment availability from storage (Hayhurst, [0009]). See §VII.B *supra*; Bhattacharjee, ¶183. 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. Bhattacharjee, ¶183.

In addition to Hayhurst's explicit teachings, POSAs knew of those benefits from background knowledge, and knew that storing segmented content units on multiple devices in a distributed fashion provided additional advantages in terms of:

- aggregate speed (*e.g.*, by allowing segments to be retrieved from multiple sources as opposed to one source); and
- efficiency/intelligence (*e.g.*, by being able to select storage device(s) that would provide the best delivery, based on availability, location, etc.).

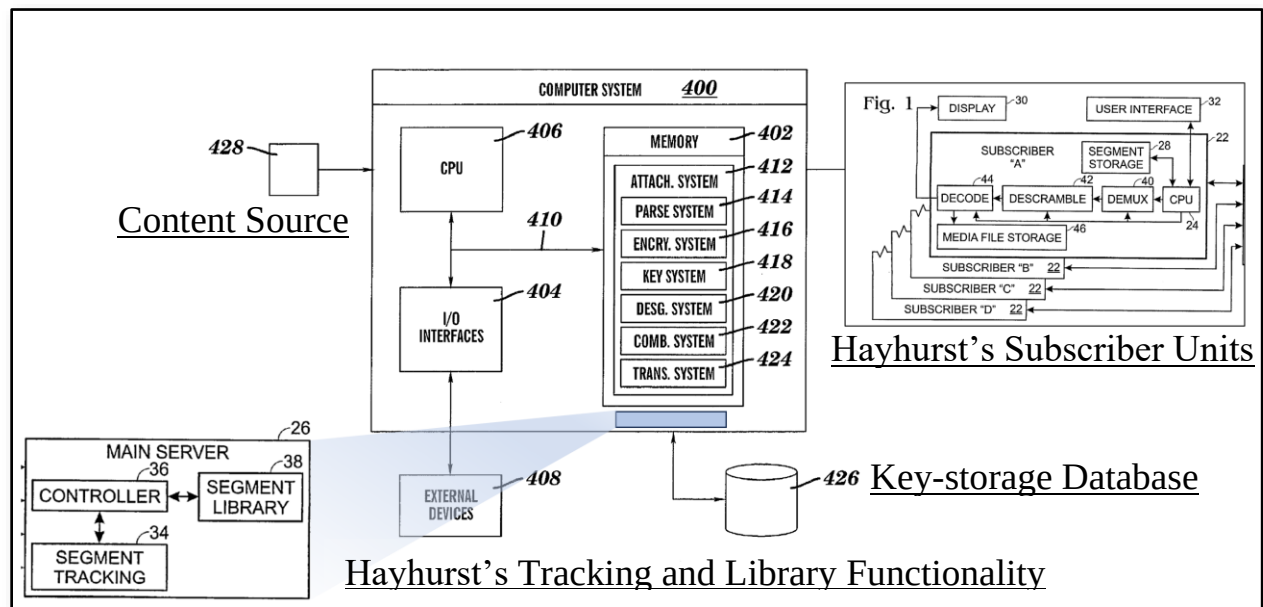
Bhattacharjee, ¶184 (citing as corroboration EX1062, [0022]-[0023]; EX1037, [0003], [0012]-[0013], [0015], [0018]-[0024]; EX1043, [0005]-[0010]; EX1050, 1:19-22, 2:31-38, 11:11-26, 16:6-20). POSAs thus had well-known reasons to incorporate Hayhurst's distributed-storage technique into Foster's system, in addition to motivation from Hayhurst itself. Bhattacharjee, ¶184.

Predictable Result. Foster-Hayhurst would have been an obvious combination of prior-art elements—combining Foster’s key-storage system and encryption method to generate encrypted data segments with Hayhurst’s method for storing encrypted data segments on multiple distributed storage devices—according to known encryption and data-storage methods, to yield the predictable result of Foster’s encrypted content units being stored on and retrievable from distributed storage devices as taught in Hayhurst. *KSR*, 550 U.S. at 416; Bhattacharjee, ¶185.

Known Technique. Foster-Hayhurst would have also been an obvious 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. *KSR*, 550 at 417-18; *Intel Corp.*, 61 F.4th at 1380-81; Bhattacharjee, ¶186.

The Foster-Hayhurst combination would have used Foster’s technique of encrypting portions of a data set with respective encryption keys that are themselves encrypted with a KEK that is stored in a key-storage system, and Hayhurst’s technique of storing the encrypted data on different storage devices. Bhattacharjee, ¶187.

One exemplary implementation of Foster-Hayhurst is depicted below (based on 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 it 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; Bhattacharjee, ¶188.



Foster-Hayhurst (exemplary implementation)

POSAs would have reasonably expected success because both references teach segmenting content for content providers and encrypting each segment, and POSAs understood that Foster's content units are individually accessible and agnostic to where and how they are stored compared to other content units, and thus

were compatible with being stored on multiple, separate storage devices (like Hayhurst’s subscriber storage devices). Bhattacharjee, ¶¶188-89; Foster, [0040]-[0041], [0061]; Hayhurst, [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”).

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. Bhattacharjee, ¶190.

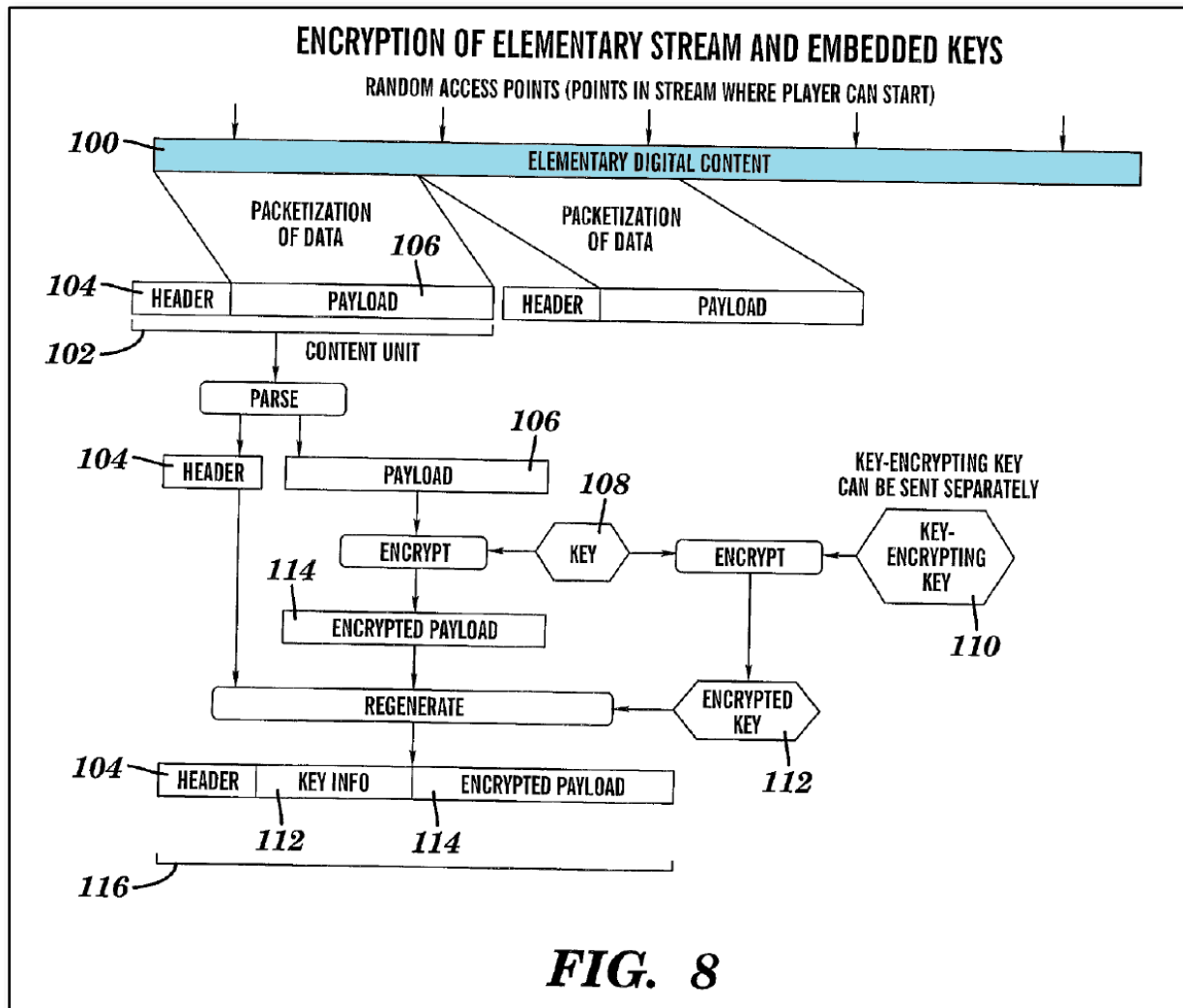
D. Claim 1:

1. 1[PRE]

Foster-Hayhurst performs a “*method for securely storing a data set*,” as claimed.

Foster-Hayhurst (§VII.C) performs Foster’s method for securing digital content, *e.g.*, “video and audio data,” by encrypting content for “storage by a receiver.” Foster, [0004], [0008], FIG. 8 (“Elementary Digital Content,” colored below), [0037] (“Content—any data such as digital image, sound or binary data deliverable from a source to a recipient”), [0050] (“content” is a “file”); Bhattacharjee, ¶¶191-92. Foster’s digital content is a “*data set*” under both parties’

constructions because POSAs understood that, *e.g.*, a video+audio data file 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” (Google; EX1064, 9-10) and “a collection of information for storage” (SFI; EX1065, 6-7). Bhattacharjee, ¶192.

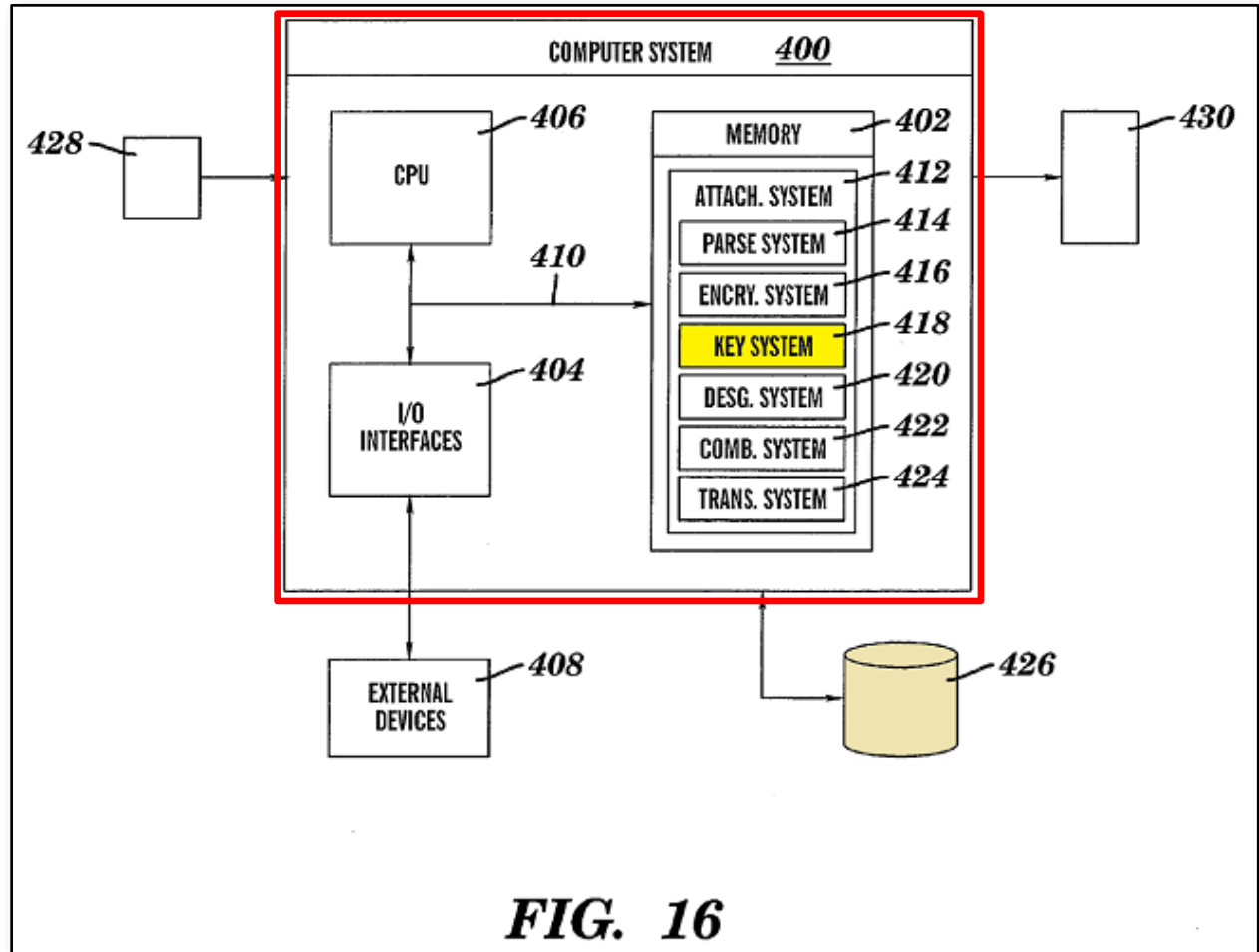


2. [1A]

Foster-Hayhurst “*receiv[es] an external key from an external storage system,*” as claimed.

Foster-Hayhurst (§VII.C) performs Foster’s encryption method and uses Foster’s key-storage system. As discussed in more detail below (§VII.D.6, [1B3] & §VII.D.7, [1C]), Foster’s encryption method includes a step of encrypting distinct title keys with a key encrypting key (KEK), which is (or alternatively would have been obvious to implement as) an “*external key*” as explained below. Bhattacharjee, ¶¶193-94. This step is illustrated in Foster’s Figure 8 (annotated below):

“use[s] the key encrypting key...to encrypt the title key[s].” Foster, [0079], [0046];
Bhattacharjee, ¶195.



“Database 426 [(color-annotated above)] provide[s] storage for” the Key Management Block (“KMB”), which is a “data structure containing” the key encryption key (KEK). Foster, [0046], [0077] (“database 426 may include...storage devices”), FIG. 16. Thus, database 426, which includes “storage devices,” is a (key) storage system. Foster, [0077]; Bhattacharjee, ¶196.

In Foster-Hayhurst, as discussed *supra* §VII.C, content data are encrypted by computer system 400 and then stored at recipients 430 (subscriber units). Database 426 does not store or encrypt the content data and, as illustrated in Figure 16, it is a distinct component from computer system 400 (red-box annotation) and from recipients 430. Bhattacharjee, ¶197. Indeed, “database 426” may be distributed across “wide area network (WAN) or a storage area network (SAN) (not shown),” and thus is separate from the computer in system 400 (color-annotated above). Foster, [0077].

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 database 426 (which stores the KEK) is an “*external storage system*” under the apparent scope SFI attributes to “*external storage system*” in its infringement contentions (EX1009, 2-3) discussed *supra* §VI.B.2.a.i. Bhattacharjee, ¶198.

When encrypting content data, Foster’s computer 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,*” as claimed. Bhattacharjee, ¶199.

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), as 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 handled by a third party system. Bhattacharjee, ¶200.

3. [1B]

As discussed 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.

Foster-Hayhurst (§VII.C) performs Foster's encryption method. As illustrated in Foster's Figure 8 (red-box annotation below), Foster generates a plurality of “payloads” by “parcel[ing]” the elementary content stream (the claimed “*data set,*” §VII.D.2 [1A]). Foster, [0064]; Bhattacharjee, ¶¶201-02.

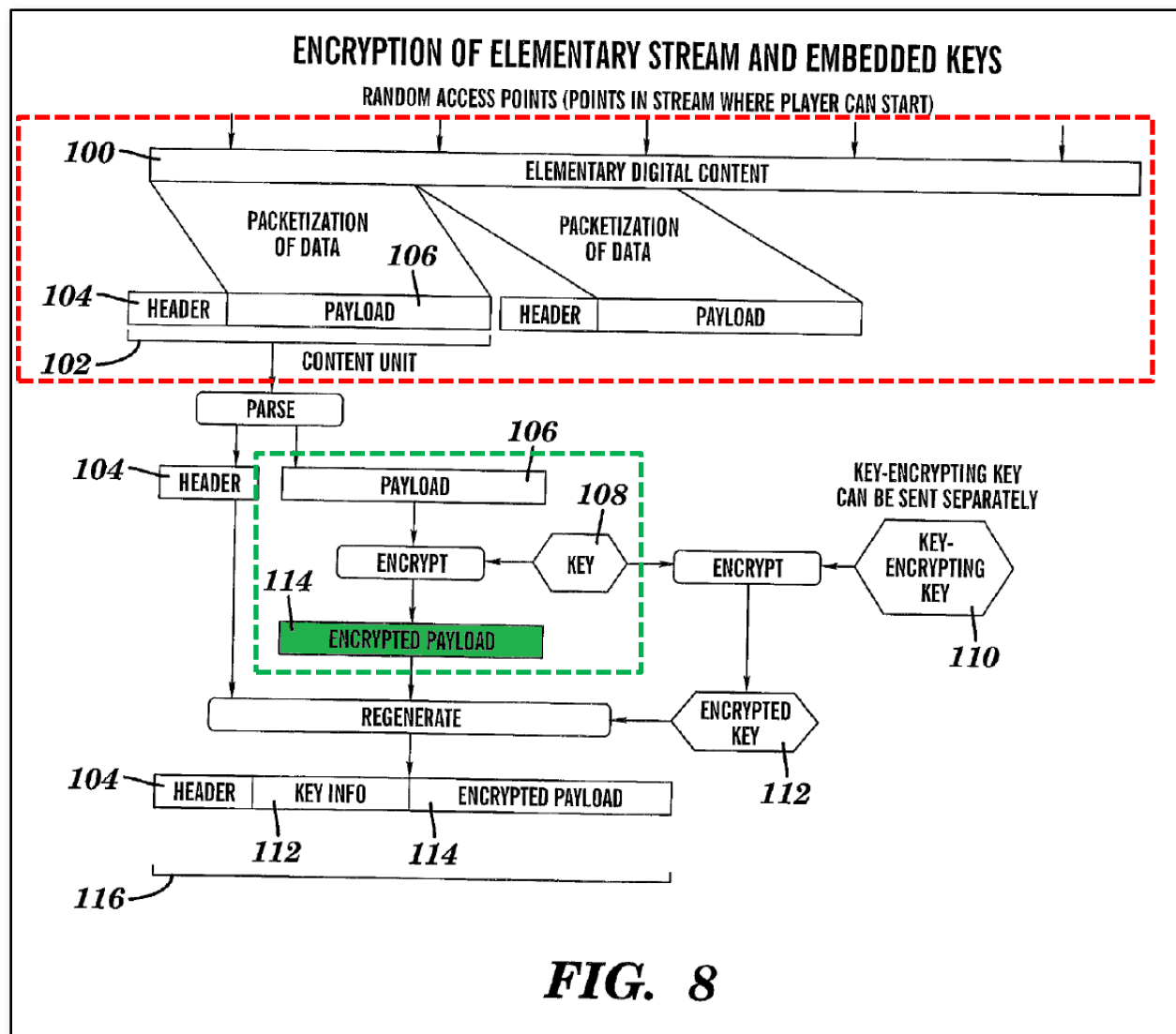


FIG. 8

As discussed further below (§VII.D.4 [1B1], §VII.D.6 [1B3]) and illustrated in Figure 8 (green-box 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 to meet SFI’s construction of the “generating” step because Foster is “creating a plurality of data chunks

[encrypted payloads] based on the data set [digital content].” EX1065, 1. Bhattacharjee, ¶203.

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, [0071]. Thus, all of the encrypted content packets (“*data chunks*”) are required in order to access (“*reconstruct*”) the entire stream (“*data set*”). Therefore, the elementary content stream (the claimed “*data set*,” §VII.D.2 [1A]) “*can be reconstructed using at least a minimum number of the plurality of data chunks*,” under SFI’s construction of “*minimum number of...chunks*,” which includes reconstructing using “the total number of data chunks.” EX1065, 13; see also EX1009, 4-5; Bhattacharjee, ¶¶204-05.

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*” that is “a predefined number of the lowest amount of the plurality of data chunks” (*e.g.*, 90 of 100 data chunks). EX1064, 13. This would have been applying a known data-distribution technique (as evidenced by, *e.g.*, Orsini, EX1050, EX1052) to a known device (Foster’s encryption device) ready for the improvement of being able to recover data from fewer than all chunks and thus protect against an inability to retrieve individual

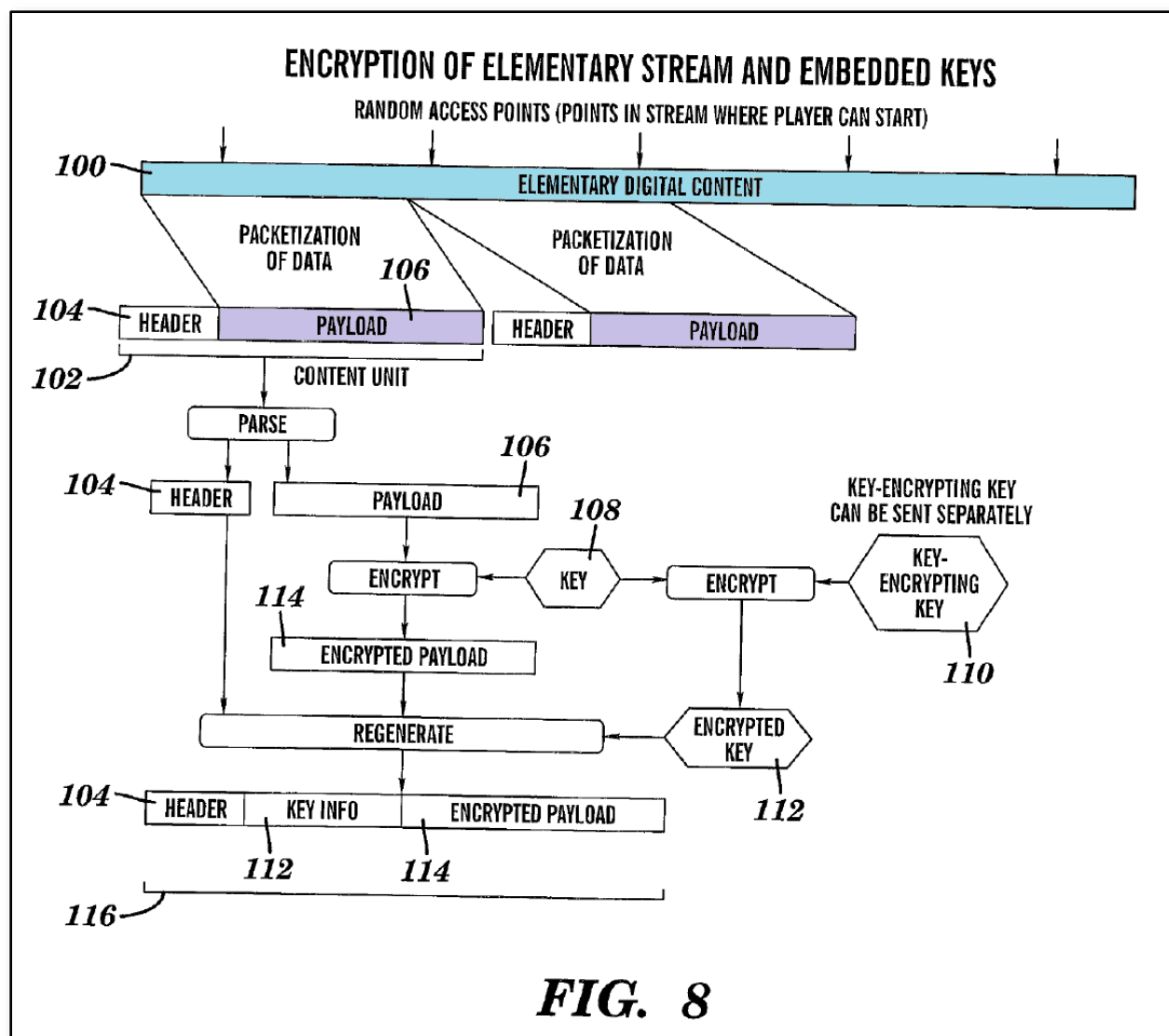
data chunks. Orsini, [0280]-[0284]; §VI.B.3 *supra*. Bhattacharjee, ¶¶206-09 (citing EX1050, 1:19-22, 2:31-38, 11:11-26, 15:39-16:20; EX1052, [0001]).

4. [1B1]

As discussed 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.

As illustrated in Foster’s Figure 8 (color-annotated below), Foster “parcel[s]” (“*distribut[es]*”) the **elementary content stream** (“*data set*”) into a plurality of “**payloads**” (“*shares*”). [0064]. Foster thus meets both parties’ proposed constructions of “*distributing the data set...*” because Foster is “splitting” (Google) and “dividing and placing” (SFI) “*the data set into a plurality of shares.*” EX1064, 2-5; EX1065, 3. Bhattacharjee, ¶¶210-11.

As Foster’s Figure 8 illustrates, each payload “*comprises less than all of the data set*,” as claimed, because Foster discloses the entire stream (“*data set*”) cannot be recovered from any one single payload. Foster, [0071]. Bhattacharjee, ¶212.

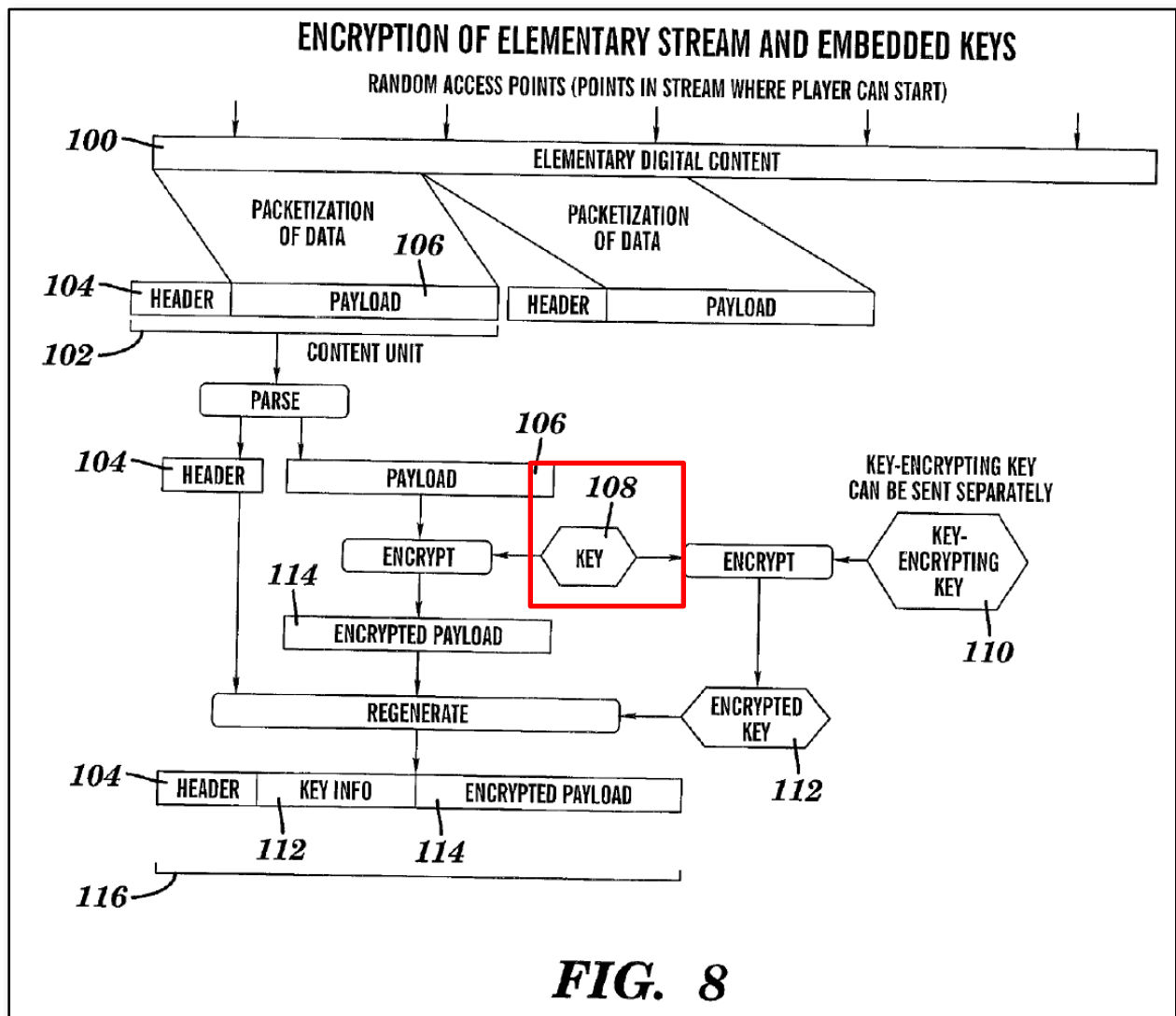


5. [1B2]

As discussed below, Foster-Hayhurst’s process for “*generating the data chunks,*” further comprises “*accessing a plurality of distinct encryption keys*” as claimed.

As illustrated in Foster’s Figure 8 (annotated below), Foster-Hayhurst’s encrypting device uses title keys (the “*plurality of distinct encryption keys*”) to

encrypt each payload (“*share*”). Foster, [0064]-[0065] (“each content packet could be encrypted with a different title key”). Thus, Foster-Hayhurst retrieves title keys, thereby “*accessing*” them (*e.g.*, from Foster’s key database 426 or a conventional key-generating/key-storage device). Bhattacharjee, ¶¶213-14; *see also* §VI.B.5 *supra*.



6. [1B3]

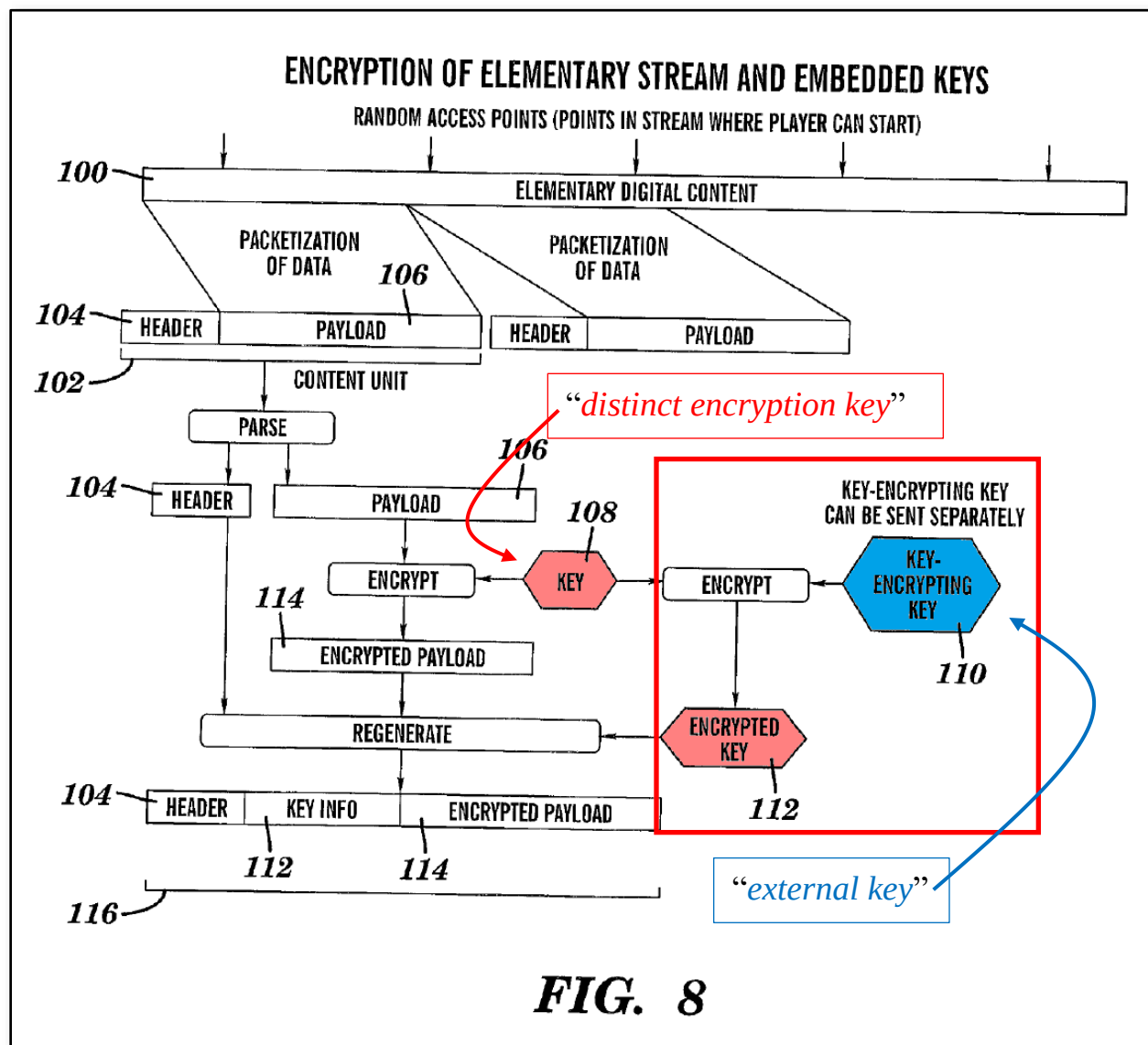
As discussed below, Foster-Hayhurst's process for "*generating the data chunks*" further comprises "*encrypting each of the shares with a respective one of the plurality of distinct encryption keys*" as claimed.

In Foster-Hayhurst, "each content unit" (FIG. 8's payload, which is a "*share*," as claimed) "could be encrypted with a **different title key**," which is "*one of the plurality of distinct encryption keys*," as claimed. Foster, [0064]-[0065], [0071]; Bhattacharjee, ¶¶215-16.

7. [1C]

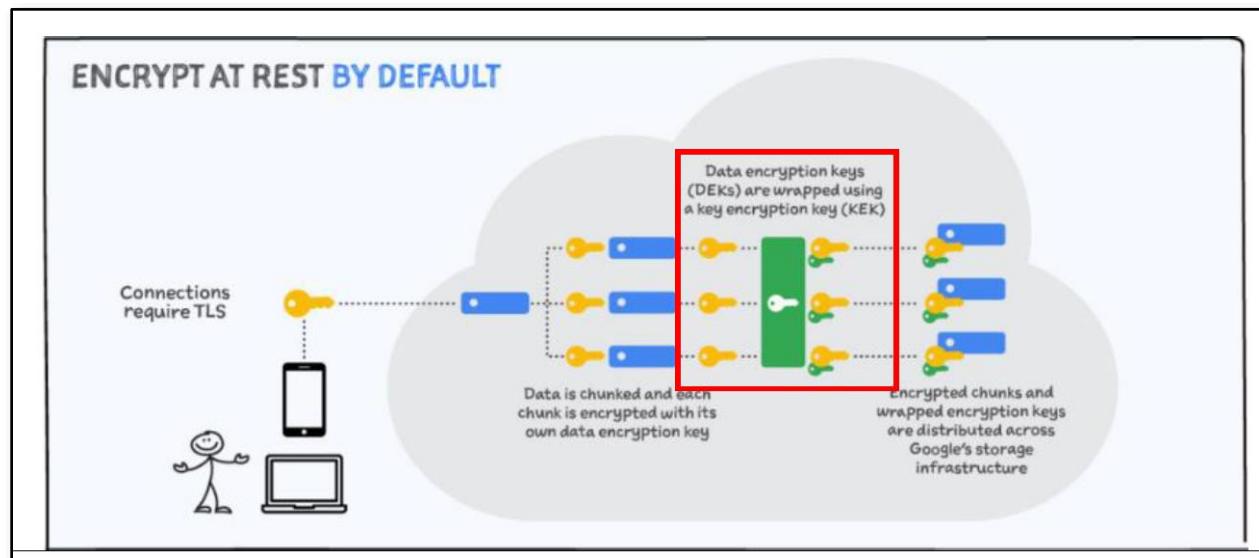
As discussed below, Foster-Hayhurst "*perform[s] an encryption operation based on the external key to further secure the plurality of data chunks*," as claimed.

Foster's Figure 8 (annotated below) illustrates encrypting each **title key** with the **KEK** (the claimed "*external key*," see §VII.D.3 [1B]), which is an encryption operation based on the KEK. Foster, [0064]-[0065]; Bhattacharjee, ¶¶217-18.



As shown below, under the scope that SFI attributes to [1C] in its district-court infringement contentions, encrypting [1B3]’s title keys with a KEK meets [1C]. See EX1009, 8-9 (“DEKs [data encryption keys] are encrypted with (wrapped

by) a key encryption key (KEK)"); §VI.B.7.b.iv. Thus, Foster meets [1C] under SFI's assertion of the claim scope. Bhattacharjee, ¶219.



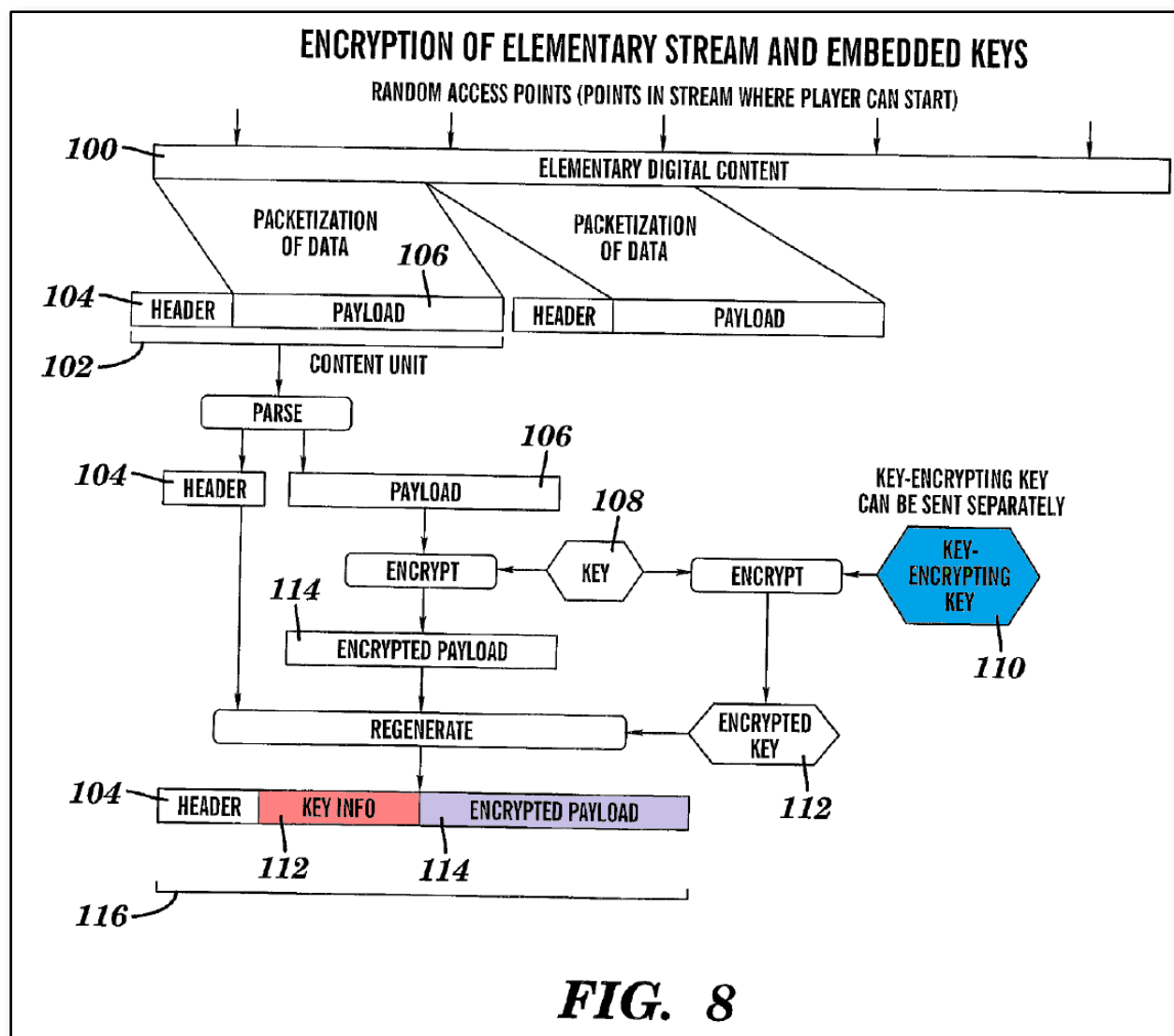
EX1009, 8

Alternatively, [1C] is met because POSAs also had motivation for (and expected success in) reordering data within each “*data chunk*” based on the “*external key*” (e.g., Foster’s KEK or a random number generated therefrom), which “*further secure[s] the plurality of data chunks,*” as claimed, for reasons discussed *supra* §VI.7.b.iii that POSAs had motivation to reorder data within Orsini’s chunks. See also *infra* §VII.G. Bhattacharjee, ¶220 (citing EX1028, EX1031, EX1032, EX1040, EX1041, EX1045, EX1054).

8. [1D]

As discussed 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.

As illustrated in Foster’s Figure 8 (color-annotated below), Foster’s method generates and stores a “processed content unit 116,” which includes three pieces of data: (1) an **encrypted payload** 114 (a “**data chunk**”); (2) “encrypted **title key** 112” (a “**distinct encryption key**”), which “has been encrypted with the **KEK**” (the “**external key**”); and (3) header 104. Foster, [0065]-[0066], [0071] (“[T]itle keys 108 and 208 are attached and transmitted with the content packets they were used to encrypt.”), [0023], FIG. 9. In the Foster-Hayhurst combination, each “processed content unit 116” is stored on a receiver at a different “recipient 430.” POSAs understood such a receiver (*e.g.*, consumer set-top box) includes a “storage device” (Foster, [0058] (“receiver may **cache**” (store) “KMBs...in **memory**” (a storage device)); also the combination’s recipients are implemented with the functionality of Hayhurst’s subscriber units, including their storage devices. §VII.C; Bhattacharjee, ¶¶221-22.



Under SFI’s claim constructions and the scope SFI 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 “stor[ing] 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 (a “*distinct encryption key*”) and corresponding

encrypted payload (a “*data chunk*”) – Foster-Hayhurst meets [1D] under SFI’s claim scope. Bhattacharjee, ¶¶223-24.

For the reasons discussed *supra* §VI.B.8, POSAs also had reason to store with each content unit metadata regarding the encryption mode and/or title key length, which is “*data indicative of*” a title key under Google’s construction. EX1064, 5-6. Bhattacharjee, ¶225 (citing EX1039).

As discussed above (§§VII.D.3-VII.D.8), Foster-Hayhurst performs limitations [1B]-[1D] in order. 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. 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. Bhattacharjee, ¶226.

E. Claim 2

Foster-Hayhurst meets claim 2 under SFI’s assertion of the claim’s scope for the reasons discussed in §VII.D.8 ([1D]). Specifically, Foster-Hayhurst stores with each “*data chunk*” (Foster’s encrypted payload) the title key ([1B3]’s “*distinct encryption keys*”) used to encrypt that data chunk. Foster, [0065]-[0066], FIGS. 8-

9; EX1009, 10-11 (alleging claim 2 is met by “storing a DEK with each data chunk”); Bhattacharjee, ¶227.

F. Claim 3

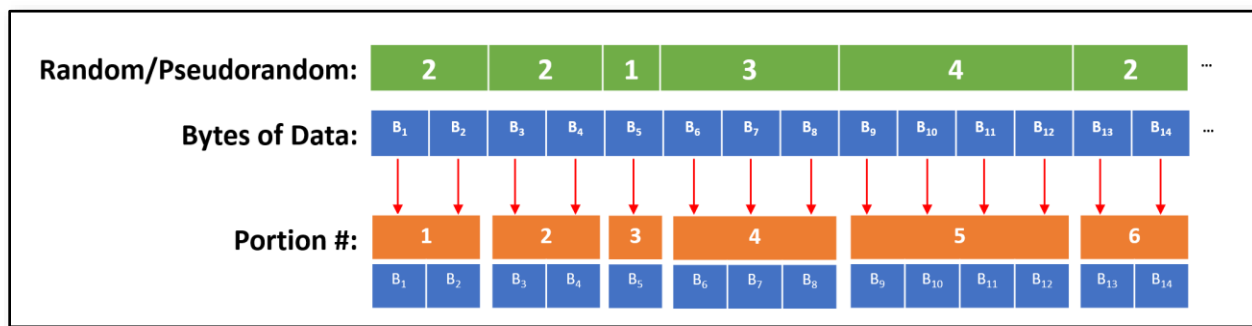
As discussed (§VI.B.8, [1D]), each encrypted payload (“*data chunk*”) in Foster-Hayhurst is stored with the title key (“*distinct encryption key*”) used to encrypt that payload. However, POSAs had reason to modify Foster to store each encrypted payload with a title key “*used to encrypt a different data chunk,*” which is “*data indicative of an encryption key,*” under SFI’s assertion of the claim scope (§VI.B.8 *supra*). POSAs knew that doing so would beneficially increase security (*e.g.*, unauthorized users could not decrypt the data without knowing which payload is stored with which key) at the potential cost of increased time to recover the data set, which was a well-understood design-choice tradeoff. Bhattacharjee, ¶228 (citing EX1061, [0059]).

This 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 payload (the “*data chunk*”) with another payload’s encryption key. *KSR*, 550 U.S. at 417; Bhattacharjee, ¶229 (citing EX1045, FIGS. 12-13, [0111]-[0116]).

G. Claim 4

Below illustrates three well-known, conventional techniques for distributing data into “shares” “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), in view of POSAs’ background knowledge evidenced by multiple references cited below.

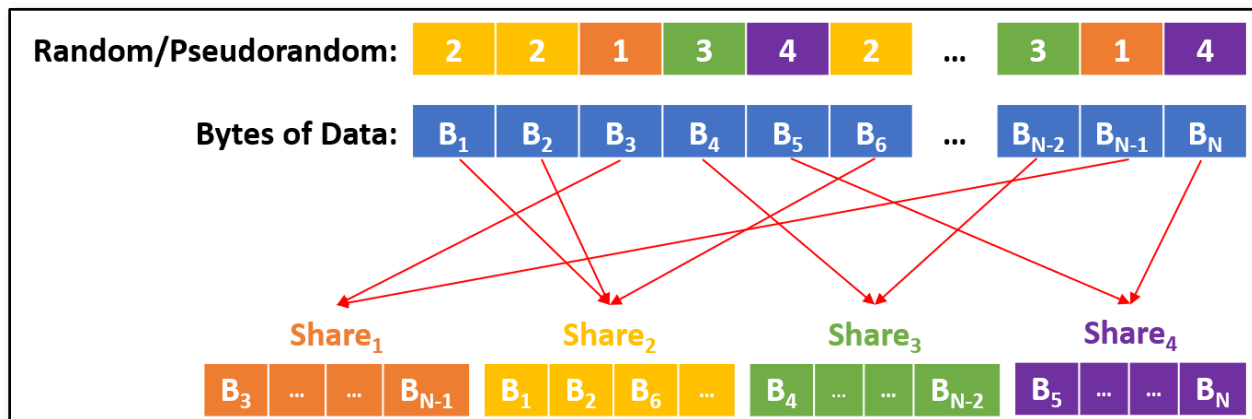
First, dividing the data set into random/pseudorandom-sized payload (“share”) portions (as illustrated below). Bhattacharjee, ¶¶230-32 (citing EX1044, [0016]-[0018], [0077]; EX1045, FIGS. 9-10, [0103]-[0108]; EX1046, [0003], [0021]; EX1028, [0015]-[0017]); Foster, [0072] (disclosing embodiment without fixed-length packets).



Second, randomly/pseudorandomly distributing a data set into payloads by rearranging data within each payload. §VI.B.7.b.iii; Bhattacharjee, ¶233 (citing EX1028, EX1044-EX1046).

Third, distributing a data set into payloads using a random/pseudorandom number as the encryption key, as illustrated below. Bhattacharjee, ¶234 (citing

EX1047, [0056]-[0058]; EX1048, [0008]; EX1045, FIGS. 9-10, [0103]-[0108]; EX1049, 10:8-15).



H. Claim 5

Foster-Hayhurst meets claim 5 for the same reason Foster-Hayhurst meets claim 4’s “*distributing...using a random or **pseudorandom technique**,*” as discussed above (§VII.G, Claim 4). Foster-Hayhurst also distributes the data set into payloads (“*shares*”) using a “*deterministic technique*” based on Foster’s disclosure of sequentially segmenting the data set. Foster, FIG. 8, [0065]; EX1001, 75:11-19 (example “deterministic algorithm” is “breaking up the original data into sequential shares”); Bhattacharjee, ¶235.

I. Claim 6

As discussed above (§VII.G, Claim 4), POSAs had reason to apply to Foster-Hayhurst well-known techniques using a random or pseudorandom (substantially random) number/key to cryptographically determine (i) payload (“*share*”) sizes,

(ii) what payload a data portion is distributed to, and/or (iii) data arrangement within each payload, any of which meets claim 6. Bhattacharjee, ¶236.

J. Claim 7

As discussed above (§VII.D.7, [1C]), Foster encrypts each encrypted payload's ("*data chunk's*") title key with a KEK ("*external key*"). Bhattacharjee, ¶237. SFI has asserted that encrypting a chunk's encryption key with a KEK is "further secur[ing]" the data chunk. EX1009, 8-9. If claim 7's "*primary data*" includes the original "*data set's*" data (*e.g.*, Foster's "elementary digital content") then Foster-Hayhurst meets claim 7 under SFI's assertion of the claim-language's scope by KEK-encrypting the title keys. Bhattacharjee, ¶238.

If Claim 7 more narrowly requires encrypting "*primary data*" directly with 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 would have been an obvious implementation of Foster-Hayhurst. POSAs had motivation and expected success in doing so to provide additional security to Foster's encrypted payloads ("*data chunks*") beyond KEK-encrypting title keys. Bhattacharjee, ¶238.

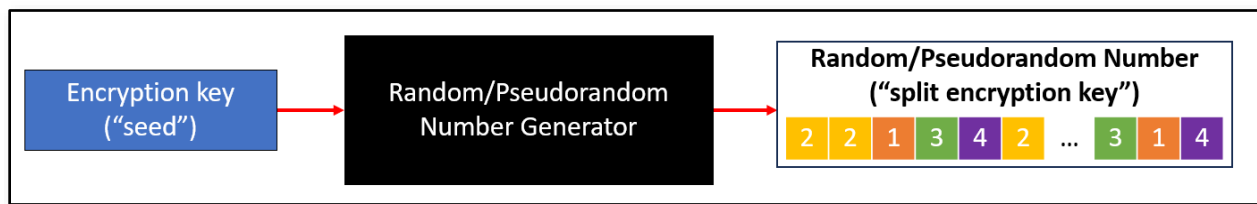
As also discussed above (§VII.G, Claim 4), POSAs had reason to use an encryption operation to distribute data into payloads ("*shares*") (*e.g.*, randomizing payload, and thus "*data chunk,*" size). To generate the distribution's

random/pseudorandom number it was conventional to use an encryption key itself or as a “seed” to a number generator. §VI.B.7.b.iv (citing corroboration). For that key, POSAs had reason to use Foster’s KEK or another conventional external key, thereby encrypting primary data with that external key. Bhattacharjee, ¶239.

K. Claim 8

As discussed (§VII.G, Claim 4), POSAs had reason to generate Foster-Hayhurst’s content units by distributing the data set into payloads (“*shares*”) using a random/pseudorandom number. For example, as discussed, that number determines the payload sizes and/or what payload receives a particular portion of the data set. See §VII.G (citing corroboration). That random/pseudorandom number meets a “*split encryption key*” because its sequence is used in the algorithm for distributing Foster-Hayhurst’s data into chunks, as discussed (§VII.I). Bhattacharjee, ¶240.

As also discussed (§VI.B.7.b.iv), it was conventional to generate random/pseudorandom numbers using an encryption key – as a so-called “seed” for a number generator. Bhattacharjee, ¶206 (citing corroboration).



Because Foster’s KEK (“*external key*”) is an encryption key used to secure the data set, POSAs had reason to also use Foster’s KEK to generate the above-discussed “*split encryption key*.” Bhattacharjee, ¶241.

Alternatively, POSAs had reason to use a conventional external encryption key (*supra* §VI.H, Claim 7) as the split-encryption-key (or its seed), to achieve the same benefits discussed above. Bhattacharjee, ¶242 (citing EX1051, [0037]).

POSAs had reason for the above-discussed keys to be stored and retrieved from Foster’s database 426 (a storage system) because database 426 was already designed to store encryption keys (*see* §VII.D.2, [1A]). 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 its seed, for the reasons discussed above in this section. Bhattacharjee, ¶243.

Accordingly, Foster-Hayhurst renders claim 8 obvious.

L. Claim 9

As discussed (§VII.D.7, [1C]), Foster-Hayhurst encrypts each payload’s (“*share’s*”) title key with a KEK (“*external key*”), which meets “*an encryption operation based on the external key to further secure the... data chunks*” under SFI’s claim scope. Thus, this encryption operation is also “*encrypting each of the shares with the external key*,” as claimed, under SFI’s apparent assertion of the claim language’s scope. Bhattacharjee, ¶244.

If claim 9 requires performing an encryption operation directly on shares, POSAs had reason to provide additional security by encrypting payloads (“*shares*”) with Foster’s KEK (or an alternative (conventional) external encryption key) before or after encrypting payloads with Foster’s distinct encryption keys. Bhattacharjee, ¶245; §VI.B.7.b *supra*.

As discussed (§VII.G, Claim 4), POSAs also had reason to distribute Foster-Hayhurst’s digital content (“*data set*”) into random/pseudorandom-sized payloads (“*shares*”) based on the “*external key*,” which is “*encrypting...the shares with the external key*.” Bhattacharjee, ¶246.

M. Claim 10

Foster-Hayhurst meets claim 10 for the same reason Foster-Hayhurst meets [1B] (§VII.D.3) and claim 9 (§VII.L). Bhattacharjee, ¶247.

VIII. NO BASIS EXISTS FOR DISCRETIONARY DENIAL

A. Section 314(a)

“[T]he PTAB will not deny institution based on *Fintiv* if,” as here, “there is compelling evidence of unpatentability.” Director’s Interim Procedure for Discretionary Denials, 5 (June 21, 2022) (“Interim Procedure”) (discussing *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (Mar. 20, 2020) (precedential)). That should conclude the *Fintiv* analysis.

If considered, the *Fintiv* factors also weigh against denial.

1. Stay Potential

Petitioner will seek a pre-institution stay. Without “specific evidence” how the court will rule, this factor is at worst neutral. *Sand Revolution II, LLC, v. Cont’l Intermodal Group–Trucking*, IPR2019-01393, Paper 24, 7 (June 16, 2020) (informative).

2. Trial Timing

The district-court trial is presently scheduled for July 2024. EX1063, 1. This factor is not dispositive especially where, as here, the petition presents strong merits. *See, e.g., PEAG LLC d/b/a JLab Audio v. VARTA Microbattery GmbH*, IPR2020-01214, Paper 8, 19-20 (Jan. 6, 2021).

3. Litigation Investment

Though there has been some activity in the district court litigation, Petitioner’s diligence in filing this Petition about eight months from being served with a complaint and about three months after SFI identified which claims would be asserted weighs against discretionary denial. *Fintiv*, 11 (expeditious filing weighs against denial); *Dish Network LLC v. Broadband iTV, Inc.*, IPR2020-01332, Paper 14, 28-29 (Jan. 27, 2021) (finding petition diligently filed within three months of receiving patent owner’s full set of asserted claims); EX1009 (2023-08-14 interrogatory response identifying asserted claims).

4. Issue Overlap

Given its early stage, the validity issues in the litigation are unknown. However, this Petition challenges 10 claims, whereas only two claims are yet asserted in the litigation. See EX1009. This weighs against denial. *Markforged Inc. v. Continuous Composites*, IPR2022-00679, Paper 7, 32-33 (Oct. 25, 2022).

5. Litigation Defendants

That Petitioner is the defendant in the district-court proceeding “is far from an unusual circumstance.” *Sand Revolution II*, 12-13. This factor is neutral.

6. Other Circumstances

The Petition’s “strong showing on the merits” weighs against denial. Interim Procedure, 3-5.

B. Section 325(d)

Under *Advanced Bionics v. Med-El*, IPR2019-01469, Paper 6 (Feb. 13, 2020) (precedential), there is no basis for discretionary denial.

1. Step 1: Art and Arguments Not Previously Considered

Ground 2 relies on Foster nor Hayhurst, which were not considered during prosecution. This fact alone warrants consideration of the petition’s merits. Only Ground 1 relies on Orsini, the issued-patent version of which was cited in an IDS and identified as “considered,” but never discussed by the Examiner, much less applied in any rejection. EX1002, 66; *see supra* §IV.C. Because Ground 2 “do[es] not involve [Orsini] at all,” any arguments under §325(d) “at best implicate only”

Ground 2, and “§ 325(d) is not sufficiently implicated” to deny institution. *ShenZhen Apaltek Co., Ltd. v. Asetek Danmark A/S*, IPR2022-01317, Paper 7, 8-10 (Feb. 6, 2023).

2. Step 2: The Office Erred Materially

If the Board finds Step 1 satisfied merely because Orsini’s issued patent was cited in an IDS, discretionary denial is still unwarranted because the Office erred materially in its prior “evaluation.” Step 2 focuses on *Becton, Dickinson* factors (c), (e), and (f) (*Advanced Bionics*, 10), which all weigh against denial.

a. Factor (c): Orsini Was Not Previously Evaluated

The Office has not previously evaluated any of this Petition’s arguments regarding claims 1-10 in view of Orsini. During prosecution, Orsini’s issued patent was only cited in an IDS. EX1002, 66. “The fact that [Orsini and its issued version] w[ere] in fact not a basis for rejection weighs against exercising discretion to deny institution under § 325(d).” *DJI Europe BV v. Textron Innovations, Inc.*, IPR2022-00453, Paper 10, 10-11 (Aug. 3, 2022).

b. Factors (e)-(f): The Office Erred

Where “the Office’s previous consideration of the art is not well developed or silent, then a petitioner may show the Office erred by overlooking something persuasive under factors (e) and (f).” *Advanced Bionics*, 10.

The Petition details how Orsini anticipates and/or renders obvious claims 1-10 (and how Foster-Hayhurst separately renders obvious claims 1-10). The Office overlooked this in allowing the claims.

IX. CONCLUSION

The Board should institute review and cancel claims 1-10.

Dated: 2023-11-22

Respectfully submitted,
Google LLC

/Elisabeth Hunt/
Elisabeth H. Hunt, Reg. No. 67,336
WOLF, GREENFIELD & SACKS, P.C.

X. 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, wherein generating the data chunks comprises:
[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.

CERTIFICATE OF SERVICE UNDER 37 C.F.R. §42.6 (E)(4)

I certify that on November 22, 2023, I will cause a copy of the foregoing document, including any exhibits or appendices filed therewith, to be served via Overnight FedEx at the following correspondence address of record for the patent:

SECURITY FIRST INNOVATIONS, LLC
C/O FARJAMI & FARJAMI LLP
26522 LA ALAMEDA AVE., SUITE 360
MISSION VIEJO, CA 92691

Date: November 22, 2023

/MacAulay Rush/
MacAulay Rush
Paralegal
WOLF, GREENFIELD & SACKS, P.C.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. §42.24, the undersigned certifies that the foregoing Petition for *Inter Partes* Review contains 13,973 words excluding a table of contents, a table of authorities, Mandatory Notices under §42.8, a certificate of service or word count, or appendix of exhibits or claim listing. Petitioner has relied on the word count feature of the word processing system used to create this paper in making this certification.

Date: November 22, 2023

/MacAulay Rush/
MacAulay Rush
Paralegal
WOLF, GREENFIELD & SACKS, P.C.