

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. IPR2024-00214

Patent No. 11,068,609

**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.....	ix
A. Real Party-In-Interest	ix
B. Related Matters	ix
1. United States Patent & Trademark Office.....	ix
2. U.S. District Court for the Eastern District of Virginia.....	x
C. Counsel and Service Information - § 42.8(b)(3) and (4)	x
I. INTRODUCTION	1
II. STANDING	2
III. UNPATENTABILITY Grounds	3
IV. THE '609 PATENT	3
A. Specification.....	3
B. Person of Ordinary Skill in the Art (“POSA”).....	5
C. Prosecution History	5
D. Challenged Claims	7
V. CLAIM INTERPRETATION	8
VI. <u>GROUND 1</u> : CLAIMS 1-10 ARE ANTICIPATED BY OR ALTERNATIVELY OBVIOUS OVER ORSINI.....	9
A. Orsini (EX1005).....	9
B. Claim 1:	17
1. 1[PRE] A method for securing data, the method comprising:	17
2. [1A] executing code by a processor to perform:	17
3. [1B] receiving a first key from a storage system,.....	18
a. Orsini discloses [1B].....	18
b. Alternatively, [1B] is obvious over Orsini.....	23
4. [1C1] generating a plurality of data chunks based on a data set, wherein each data chunk of the plurality of data chunks comprises less than an entirety of data of the data set, and.....	26
5. [1C2] wherein the data set can be reconstructed using at least a minimum number of the plurality of chunks;	28

6.	[1D] encrypting each respective data chunk of the plurality of data chunks with a respective second key, wherein each of the respective second keys are distinct from each other;	29
7.	[1E] performing a cryptographic operation based on the first key to further secure the plurality of data chunks; and	30
a.	Orsini discloses [1E]	30
i.	Performing encryption steps in different order.....	30
ii.	Repeating the encryption steps.	33
b.	Alternatively, [1E] is obvious over Orsini	36
i.	POSAs had reason to re-encrypt data chunks with a first encryption key.	36
ii.	POSAs had reason to re-encrypt data chunks with a group key.	37
iii.	POSAs had reason to reorder data within each chunk based on an encryption key.....	38
iv.	POSAs had reason to encrypt [1D]’s “ <i>respective second keys</i> ” with a key-encryption key.....	40
8.	[1F] storing, in a memory coupled to the processor, at least one data chunk of the plurality of data chunks with data indicative of at least one of the distinct encryption keys on at least one storage device.	42
C.	Claim 2: The method of claim 1, further executing the code by the processor to perform: storing with each data chunk of the plurality of data chunks data indicative of at least one of the distinct encryption keys.	45
D.	Claim 3: The method of claim 1, further executing the code by the processor to perform: storing with each data chunk of the plurality of data chunks data indicative of a distinct encryption key used to encrypt that data chunk.	46
E.	Claim 4: [1C1]’s generating...comprises using a random technique or pseudorandom technique.....	49
F.	Claim 5: [1C1]’s generating...comprises a substantially random distribution of a subset of the data set.....	50

G.	Claim 6: [1C1]’s generating...comprises: generating a split encryption key based on the first key; and distributing the data set based on the split encryption key.	51
H.	Claim 7: The method of claim 1, further executing the code by the processor to perform: encrypting each of the chunks of the plurality of chunks with the first key.	53
I.	Claim 8: The method of claim 7, wherein the data set is recoverable using at least a minimum number of the plurality of data chunks and the first key.	54
J.	Claim 9: Claim 1’s first key is an AES key; Claim 10: Claim 1’s first key is an DES key.	54
VII.	<u>GROUND 2: CLAIMS 1-10 ARE UNPATENTABLE OVER FOSTER</u>	56
A.	Foster (EX1006).....	56
B.	Claim 1:	60
	1. 1[PRE]	60
	2. [1A].....	62
	3. [1B]	63
	4. [1C1]	65
	5. [1C2]	67
	6. [1D].....	68
	7. [1E]	70
	8. [1F].....	72
C.	Claim 2	78
D.	Claim 3	79
E.	Claim 4	80
F.	Claim 5	81
G.	Claim 6	82
H.	Claim 7	83
I.	Claim 8	84
J.	Claims 9-10	85

VIII.	NO BASIS EXISTS FOR DISCRETIONARY DENIAL	86
A.	Section 314(a)	86
1.	Stay Potential	86
2.	Trial Timing.....	86
3.	Litigation Investment.....	86
4.	Issue Overlap	87
5.	Litigation Defendants	87
6.	Other Circumstances.....	87
B.	Section 325(d)	87
1.	Step 1: The Petition Advances Art and Arguments Not Previously Considered	88
2.	Step 2: The Office Erred Materially	88
a.	Factor (c): Orsini Was Not Previously Evaluated	88
b.	Factors (e)-(f): The Office Erred.....	89
IX.	CONCLUSION.....	89
X.	APPENDIX: CLAIM LISTING.....	92

TABLE OF AUTHORITIES

Cases

<i>10X Genomics, Inc. v. Bio-Rad Labs.</i> , IPR2020-00086, Paper 8 (Apr. 27, 2020).....	9
<i>Advanced Bionics v. Med-El</i> , IPR2019-01469, Paper 6 (Feb. 13, 2020).....	87, 88, 89
<i>Apple Inc. v. Fintiv Inc.</i> , IPR2020-00019, Paper 11 (Mar. 20, 2020)	86, 87
<i>Becton, Dickinson & Co. v. B. Braun Melsungen AG</i> , IPR2017-01586, Paper 8 (Dec. 15, 2017).....	88
<i>Dish Network LLC v. Broadband iTV, Inc.</i> , IPR2020-01332, Paper 14 (Jan. 27, 2021).....	87
<i>DJI Europe BV v. Textron Innovations, Inc.</i> , IPR2022-00453, Paper 10 (Aug. 3, 2022)	88
<i>Intel Corp. v. PACT XPP Schweiz AG</i> , 61 F.4th 1373 (Fed. Cir. 2023)	37, 77
<i>KSR Int’l Co. v. Teleflex Inc.</i> , 550 U.S. 398 (2007).....	passim
<i>Markforged Inc. v. Continuous Composites</i> , IPR2022-00679, Paper 7 (Oct. 25, 2022)	87
<i>Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.</i> , 868 F.3d 1013 (Fed. Cir. 2017)	8
<i>PEAG LLC d/b/a JLab Audio v. VARTA Microbattery GmbH</i> , IPR2020-01214, Paper 8 (Jan. 6, 2021).....	86
<i>Sand Revolution II, LLC, v. Cont’l Intermodal Group–Trucking</i> , IPR2019-01393, Paper 24 (June 16, 2020).....	86, 87
<i>Uber Techs., Inc. v. X One, Inc.</i> , 957 F.3d 1334 (Fed. Cir. 2020)	26

Statutes

35 U.S.C. §102.....	3
35 U.S.C. §103.....	3
35 U.S.C. §314(a)	86
35 U.S.C. §325(d)	87, 88

Regulations

37 C.F.R. §42.100(b)	8
37 C.F.R. §42.104(a).....	2

APPENDIX LISTING OF EXHIBITS

Exhibit	Description
1001	U.S. Patent No. 11,068,609
1002	Prosecution History of U.S. Patent No. 11,068,609
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	RESERVED
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. 11,068,609
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	RESERVED
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	RESERVED
1021	RESERVED
1022	U.S. Patent Application Publication No. 2002/0126850
1023	U.S. Patent No. 5,812,671
1024	RESERVED
1025	RESERVED
1026	RESERVED
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

Exhibit	Description
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	RESERVED
1038	RESERVED
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	U.S. Patent No. 6,363,481
1043	RESERVED
1044	U.S. Patent Application Publication No. 2003/0191938
1045	U.S. Patent Application Publication No. 2003/0028493
1046	U.S. Patent Application Publication No. 2003/0016820
1047	U.S. Patent Application Publication No. 2002/0042859
1048	U.S. Patent Application Publication No. 2003/0188153
1049	PCT Application Publication No. WO 2004/057461
1050	U.S. Patent No. 7,529,834
1051	U.S. Patent Application Publication No. 2006/0034457
1052	U.S. Patent Application Publication No. 2005/0256972
1053	U.S. Patent Application Publication No. 2004/0049700
1054	U.S. Patent No. 5,675,653
1055	U.S. Patent Application Publication No. 2006/0062384
1056	U.S. Patent No. 6,490,353
1057	U.S. Patent Application Publication No. 2003/0044020
1058	U.S. Patent No. 7,092,524
1059	PCT Application Publication No. WO 2005/024686
1060	U.S. Patent Application Publication No. 2006/0206889
1061	U.S. Patent Application Publication No. 2006/0059347
1062	RESERVED
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

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. 11,068,609 issued is a Continuation of U.S. Patent Application No. 16/127,077 (U.S. Patent No. 10,452,854), filed on September 10, 2018, which 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,

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.

which claims priority to U.S. Provisional Application No. 60/738,231, filed November 18, 2005.

The following U.S. patent application claims the benefit of priority to U.S. Patent No. 11,068,609: 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 (pro hac vice 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. 11,068,609 (“the ’609 patent”) (EX1001).

The ’609 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, 60:62-61:32. It likewise relies on conventional techniques for splitting data into portions. EX1003 (hereinafter “Bhattacharjee”), ¶¶27-30.

Beyond conventional encryption and data-splitting operations, the challenged method claims use a “storage system” to store a “first 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 (with a “second key”). But re-encrypting data and using an encryption 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. Bhattacharjee, ¶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 '609 patent. Like the '609 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 encryption keys received from storage systems. The challenged claims are unpatentable over Orsini. Bhattacharjee, ¶31.

Ground 2 demonstrates that claims 1-10 are also unpatentable over Foster (EX1006). 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 to a storage device. Bhattacharjee, ¶32.

II. STANDING

Petitioner certifies that the '609 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		Claims	Basis
1	Orsini (EX1005)	1-10	§§102/103
2	Foster (EX1006)	1-3, 7-8	§§102/103
		4-6, 9-10	§103

Orsini and Foster are each prior art under pre-AIA 35 U.S.C. §102(b) at least because each published more than one year before the '609 patent's earliest-alleged priority date of November 18, 2005. Orsini and Foster were not cited during prosecution.

IV. THE '609 PATENT²

A. Specification

The '609 patent relates to techniques for securing electronic data from unauthorized access. 1:27-28. By 2005, individuals and businesses were storing and transmitting an ever-increasing amount of sensitive information over computer networks. 1:32-39. 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

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

possess the key(s) to the encryption can decrypt the data into a useable format.” 1:48-52. 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). 1:64-2:13; 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 encryption keys remain secured on the cryptographic servers and are not given to users. 8:66-9:6. According to the patent, “[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 it has been parsed/split/separated), and **[3]** storing the “parsed, split, and/or separated data that has been encrypted in one location or in multiple locations.” 2:45-58. The

patent says that steps [1] and [2] can be “repeated” such that the data set is parsed/split/separated and/or encrypted multiple times. 2:52-55; Bhattacharjee, ¶35.

As seen *infra* §IV.D, the challenged method claims recite performing the patent’s three basic encryption steps and further recite using a “first [encryption] key” to perform an additional “cryptographic operation” to further secure the data chunks after the data set has already been parsed and encrypted. That is, the “first key” can be used to “post encrypt” the data set after it has first been parsed and encrypted. 79:8-12; *see also* FIG. 36 (step 3619). The patent ascribes no advantages to using “first” keys or to receiving such keys from storage systems. Bhattacharjee, ¶36.

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

A POSA at the time of the ’609 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

In a preliminary amendment, SFI filed ten method claims that were substantially like the issued claims and ten corresponding system claims. EX1002, 90-102. After all claims were rejected for non-statutory double patenting over SFI’s

U.S. Patent No. 10,452,854 (the “’854 patent,” which Petitioner is challenging in co-pending IPR2024-00215), SFI cancelled the system claims and amended the method claims to recite performing the claimed steps by “executing code by a processor” and “storing” the encrypted data “in a memory coupled to the processor.” EX1002, 55-58, 70-77. After SFI filed a terminal disclaimer, the claims were allowed without comment by the examiner. EX1002, 32, 42-58.

A certificate of correction for dependent claims 6-10 was subsequently granted to replace “external key” with claim 1’s “first key.” EX1002, 11-16. Thus, as corrected, no issued claim recites an “external key,” which was likely a holdover from the ’854 patent’s claims, which resulted in the double-patenting rejection discussed above. Indeed, the principal distinction between the claims of the two patents is that the ’854 patent’s claims recite “*receiving an **external** key from an **external** storage system*” whereas the ’609 patent’s claims recite “*receiving a **first** key from a storage system.*”³ Bhattacharjee, ¶¶42-43.

³ All emphases herein are added unless otherwise indicated.

D. Challenged Claims

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

As discussed *infra* (§VI.B.2, §VII.B.2), limitation [**1A**] recites a processor “executing code” to perform the claimed “method for securing data” recited in limitations [**1B**]-[**1F**]. Limitations [**1C1**]-[**1C2**] (§§VI.B.4-VI.B.5, §§VII.B.4-VII.B.5) and [**1D**] (§VI.B.6, §VII.B.6) generate “data chunks”⁴ based on the data set and then encrypt the individual chunks with “respective second [encryption] keys” that are “distinct from each other.” Bhattacharjee, ¶¶44-45. In turn, limitations [**1B**] (§VI.B.3, §VI.B.3) and [**1E**] (§VI.B.7, §VII.B.7) use a “first [encryption] key” to “perform[] a cryptographic operation” that “further secure[s]” the data chunks, and limitation [**1F**] (§VII.B.8, §VII.B.8) stores at least one data chunk “in a memory

⁴ The claims of the ’609 patent and ’854 patent (challenged in IPR2024-00215) both recite “*data chunks*,” but the term is used differently in each claim set. The ’609 patent generates “*data chunks*” based on a “*data set*” and encrypts “*data chunks*” with respective encryption keys; the ’854 patent distributes a “*data set*” into “*shares*” and the “*data chunks*” are generated by encrypting each “*share*” with a respective encryption key. In other words, the ’609 patent’s claimed “*data chunks*” are what the ’854 patent’s claims call “*shares*.” Bhattacharjee, ¶¶44.

coupled to the processor...on at least one storage device” and storing with each data chunk “data indicative” of at least one of [1D]’s “respective second key[s].” As shown in Grounds 1-2, claim 1’s steps are disclosed in and/or would have been obvious over Orsini (Ground 1) and separately over Foster (Ground 2). Bhattacharjee, ¶¶45-46.

The dependent claims recite conventional data-encryption. For example, dependent claims 4-5 recite methods for parsing data into the data chunks, *e.g.*, using a random or substantially random technique. Dependent claims 6-10 recite methods for splitting an encryption key, distributing the data based on that split key, encrypting and recovering data chunks, and using keys based on the prior-art encryption standards AES and DES. 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, ¶47.

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

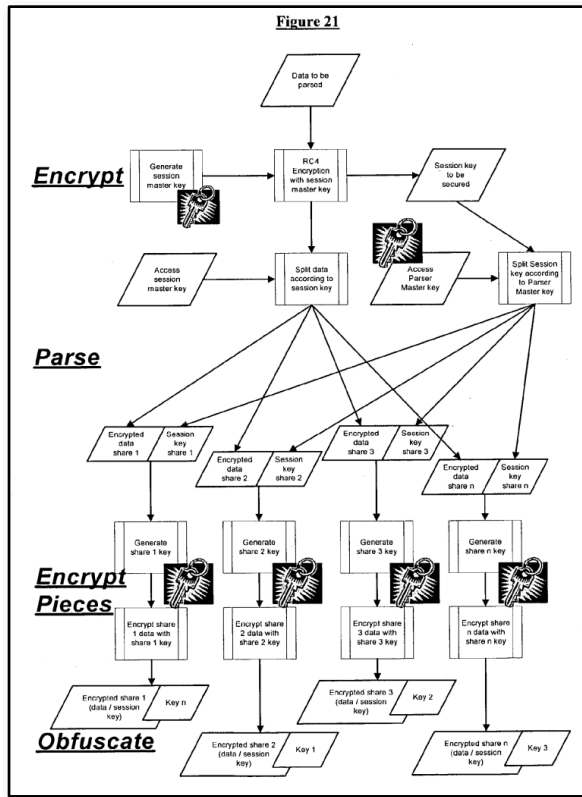
competing proposed constructions of various claim terms. EX1064-EX1065. As discussed in detail where relevant within the Grounds below, this Petition presents alternative mappings of the prior-art disclosures to both parties' proposed constructions of disputed terms, thus demonstrating unpatentability 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 BY OR ALTERNATIVELY OBVIOUS OVER ORSINI

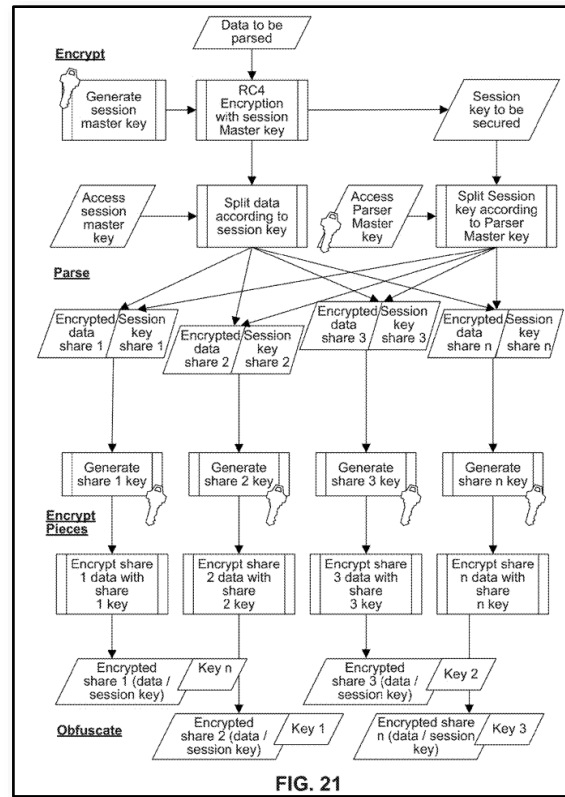
A. Orsini (EX1005)

The '609 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, 1:1-65:55, FIGS. 1-25; *see also* EX1010 (redline comparison); Bhattacharjee, ¶¶49-51.

Like the '609 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 '609 patent's Figure 21 (below right) except for formatting differences. Orsini, [0293].



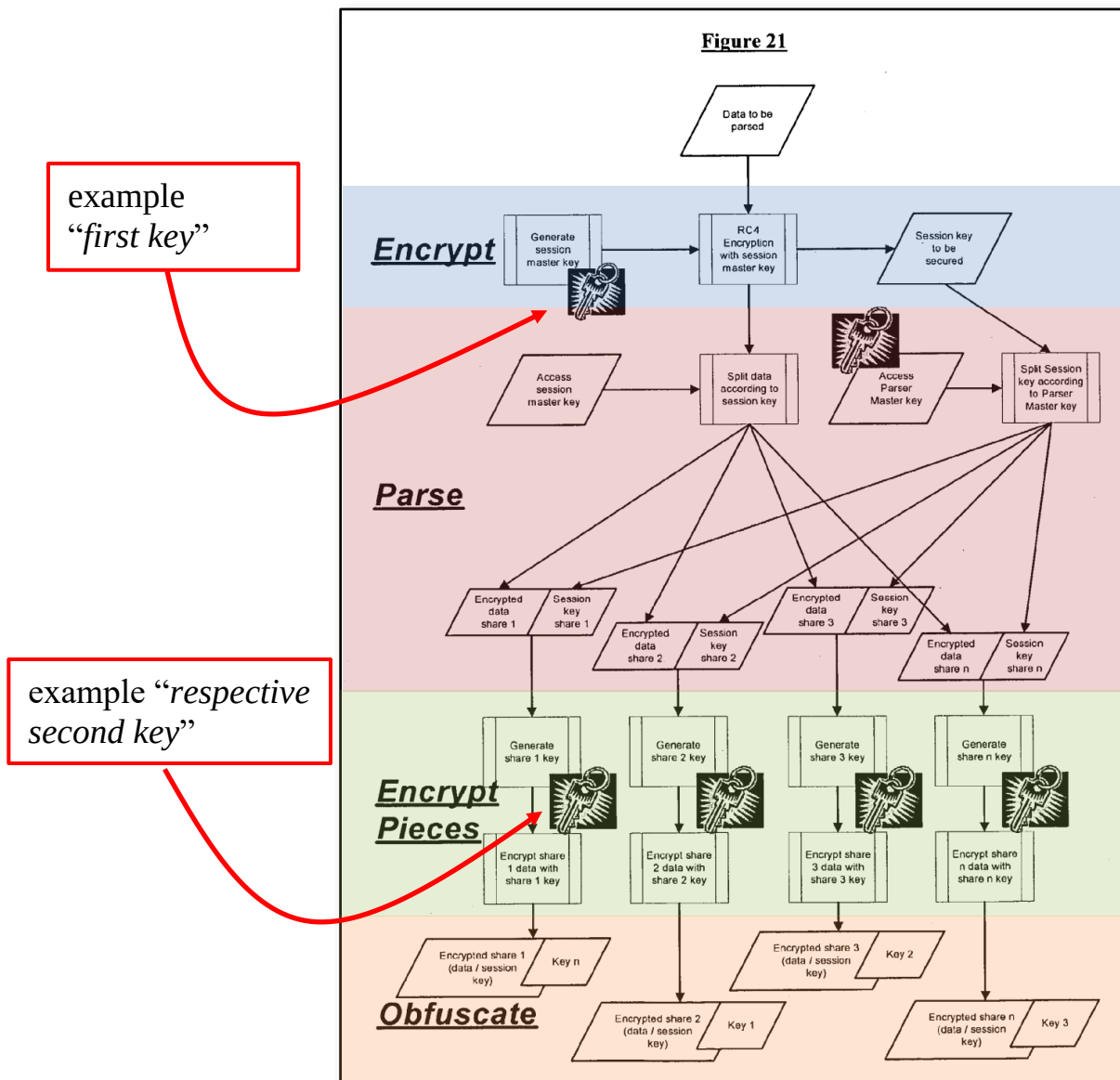
Orsini



'609 Patent

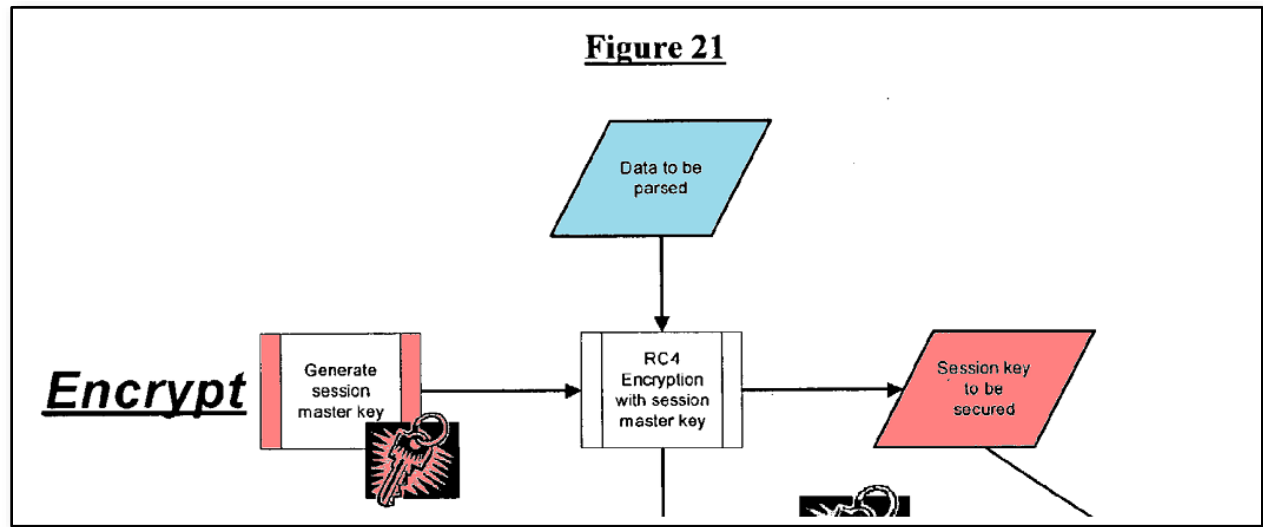
Bhattacharjee, ¶52.

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 [1B]’s “*first*,” *infra* §VI.B.3) and multiple “share keys” (which meet limitation [1D]’s “*respective second keys*,” *infra* §VI.B.6):



Bhattacharjee, ¶53.

In Orsini’s “encrypt” stage (color-annotated below), a data set (blue) is encrypted with a “session master key” (red) (Orsini, [0294]-[0295]).⁵

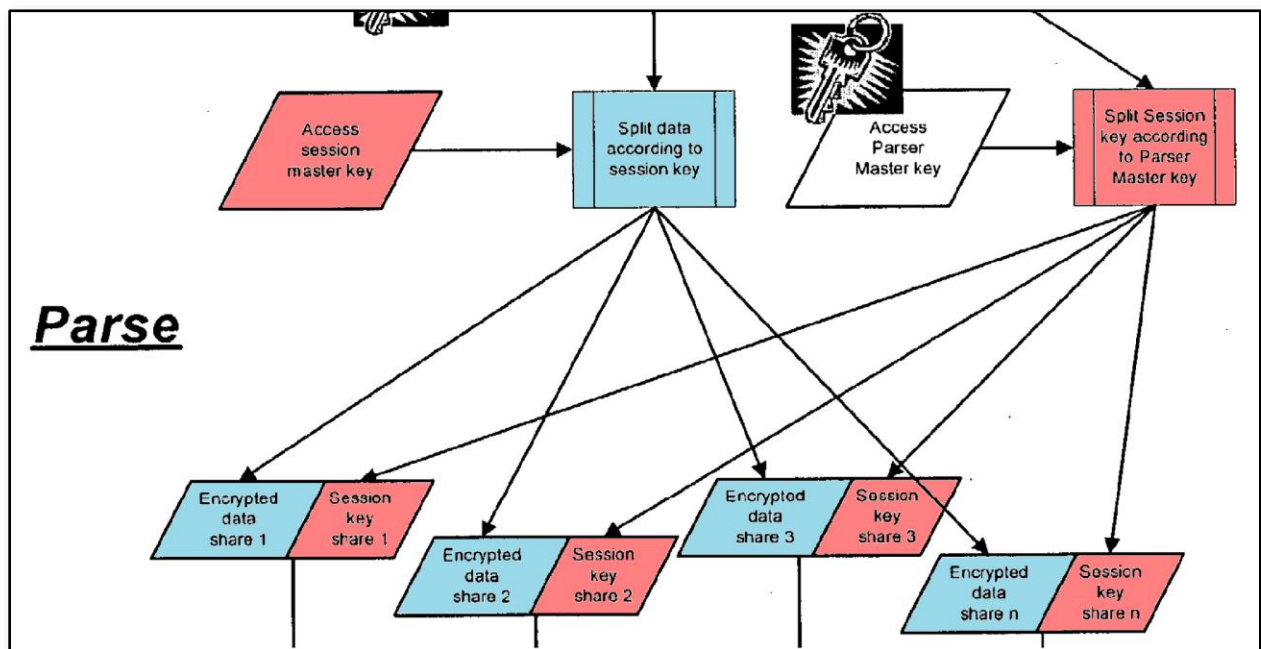


The encryption with the session master key can use any “well known” encryption algorithm.⁶ Orsini, [0371], [0294]-[0295]. The session master key can be either stored on the server performing the encryption or accessed from another source. Orsini, [0293] (discussing “external session key management or secure internal storage of session keys”); Bhattacharjee, ¶54.

⁵ “Master keys” are used to derive other encryption keys. Bhattacharjee, ¶54.

⁶ Figure 21’s first stage uses RC4 encryption—commercialized by third-party “RSA Security, Inc.”—however, Orsini states a “wide variety of encryption methodologies are suitable.” Orsini, [0335], [0371], [0378]; EX1001, 59:37-39, 61:16-46 (same).

Orsini’s “parse” stage splits the encrypted data set “into four shares or portions of parsed data according to the pattern of the session master key.” Orsini, [0296]. The session master key itself can be split “according to the pattern of the Parser Master Key,” which is yet another encryption key. Orsini, [0297], [0391]-[0392]. Each portion of the split encrypted data (blue “data shares” below) is then paired with a portion of the split session key (red “key shares” below):

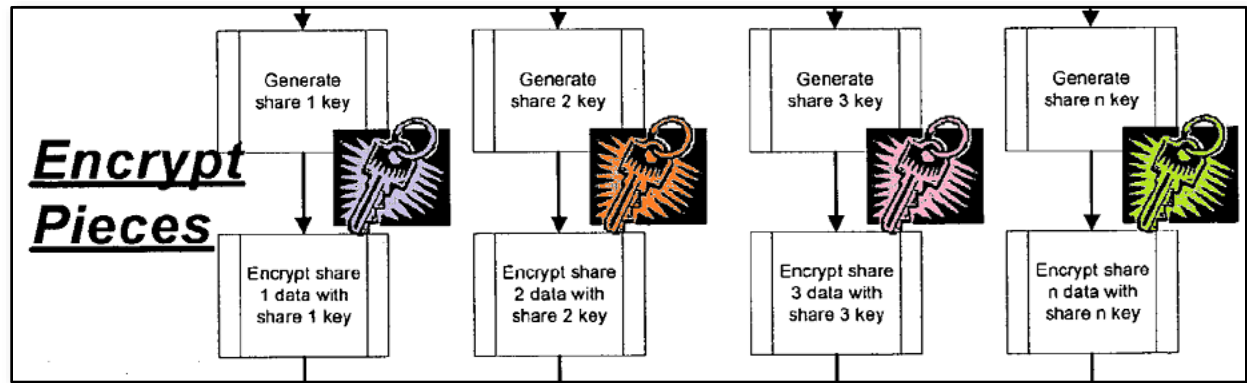


Bhattacharjee, ¶55.

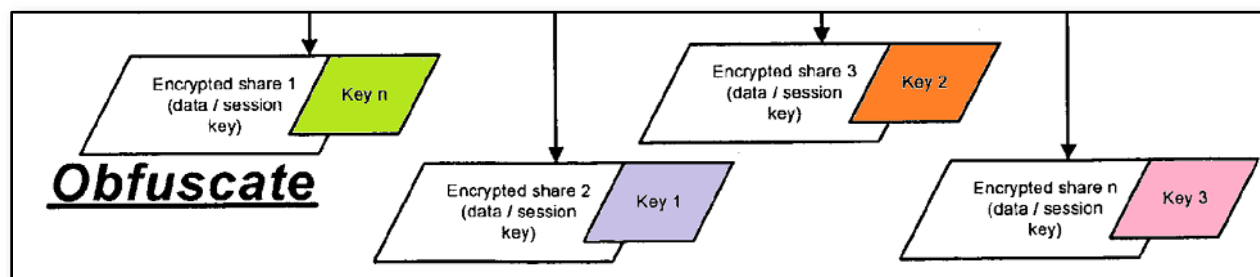
.....

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), resulting in an encrypted share (which meets [1C1]’s “*data chunk*,” §VI.B.4 *infra*). Each encrypted share includes a portion of the (encrypted) original data set and a portion

of the original session master key. Orsini, [0298]-[0299], [0395]-[0396]; Bhattacharjee, ¶56. This is depicted below:



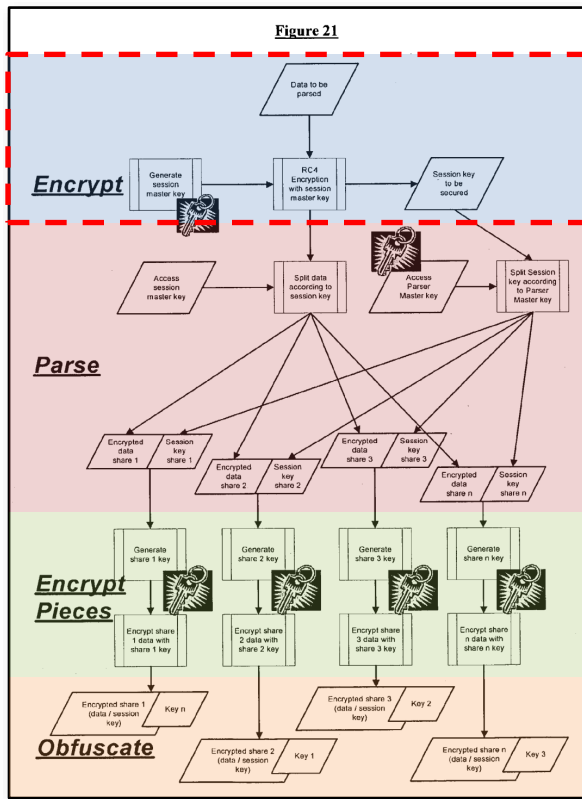
Finally, in the “obfuscate” stage, each encrypted share is paired with a share key that encrypted a *different* share (e.g., encrypted share 2 is paired with share 1 key, etc.). Orsini, [0299]. The different pairs of encrypted share and share key can then be stored, for example, in different data depositories. Orsini, [0299], [0379]; Bhattacharjee, ¶57. This is depicted below:



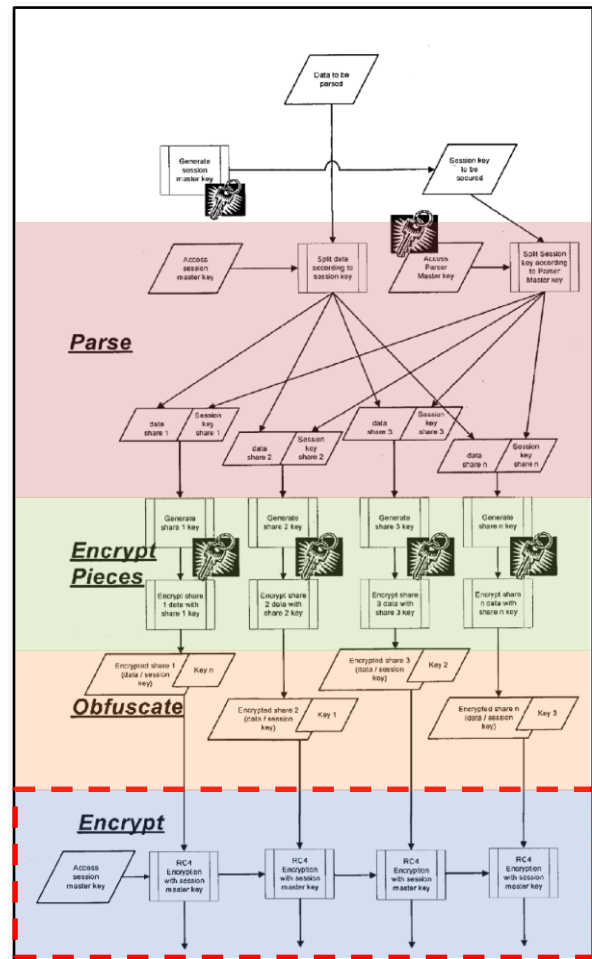
The '609 patent's claim 1 performs the same encryption steps illustrated in Orsini's Figure 21 but does not require (or exclude) encrypting the data set **before** splitting it into shares. Instead, claim 1 recites an encryption step with an encryption key (specifically, a "*first key*") that occurs **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." Orsini, [0301]; Bhattacharjee, ¶58.

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:

Original



Different-Order Version



Bhattacharjee, ¶¶59-61.

B. Claim 1:

1. 1[PRE] A method for securing data, the method comprising:

Orsini discloses a “*method for securing data*,” as claimed.

For example, Orsini discloses “a method for ***securing*** virtually ***any type of data*** from unauthorized access or use,” and a step in the method is “***storing...data that has been encrypted***.” Orsini, [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, ¶¶62-63; *see also infra* §§VI.B.2-VI.B.8 (discussing further Orsini’s method for securing data).

2. [1A] executing code by a processor to perform:

In an exemplary embodiment, Orsini’s method is executed by a computer system (*e.g.*, a server) that executes “a parser program” (which comprises “*code*”) to perform “data splitting, encryption and decryption[.]” Orsini, [0278]; *see also* [0013] (“This method may be incorporated into...***any computer, server, engine*** or the like, that is capable of ***executing*** the desired steps of the method.”); Bhattacharjee, ¶64.

POSAs understood Orsini’s “computer system” (Orsini, [0287], [0013]) comprises at least one processor for executing the parser-program code (*e.g.*, “source code,” Orsini, [0288]). Bhattacharjee, ¶65.

Executing the parser program's code using a processor would have also been the obvious combination of prior art elements (a conventional processor for a computer system) according to known computer-system techniques to yield the predictable result of performing encryption techniques with a processor executing code. *KSR*, 550 U.S. at 417; Orsini, [0063] (user system 105 comprises "a conventional general-purpose computer having one or more microprocessors"); Bhattacharjee, ¶66 (citing EX1042, 8:37-44 (disclosing systems for encryption and storage using "general purpose computer systems" including those "based on processors available from," e.g., Intel)).

3. [1B] receiving a first key from a storage system,

As discussed below, Orsini discloses "*receiving a first key from a storage system*," as claimed. 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 [1B].

a. Orsini discloses [1B]

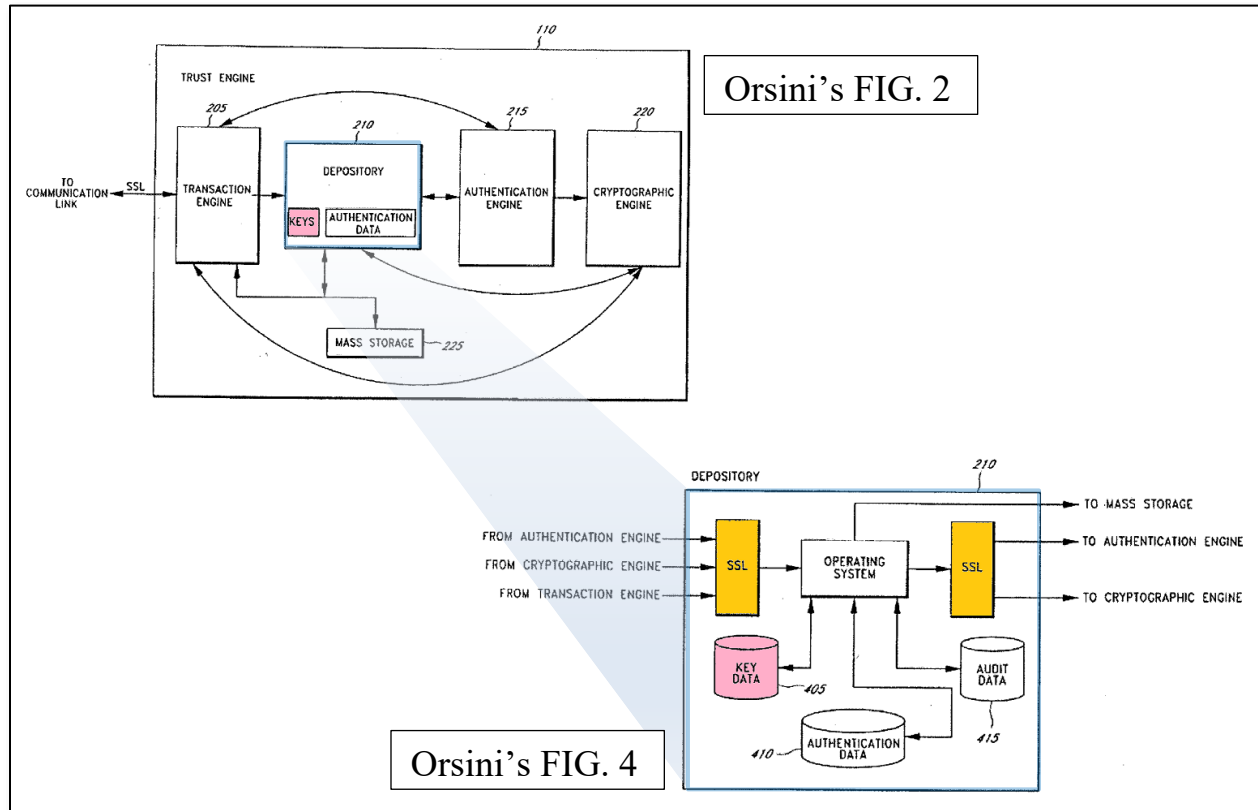
Orsini's Figure 21 shows its parser software generating the session master key without specifying at what physical device the generating occurs. Even if some embodiments generate the session master key at the encryption device without receiving it from a storage system, Orsini discloses embodiments where its encryption techniques "may be implemented with external *session key management*

or secure internal **storage of session keys.**” Orsini, [0293], *see also* [0329] (“passing the Session Master Key...to the calling program for **external management**”), [0381]-[0386], [0389]-[0394] (managing encryption keys using a “standard key stores, [or] custom key stores”), [0399]-[0402], FIGS. 1-7; Bhattacharjee, ¶¶67-68.

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, ¶69. Thus, POSAs understood that instead of the session master key being generated by the encrypting device, the key can be created, distributed, and stored by a session key management system for later use by the system implementing Orsini’s encryption methods. Bhattacharjee, ¶69.

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.” Orsini, [0034], [0036], [0066]; *see also* [0063]-[0113]. Orsini’s encryption method discussed below (§§VI.B.4-VI.B.8) is implemented in cryptographic engine 220. Orsini, [0090] (“cryptographic engine 220 [] include[s] a data splitting module”), [0278] (“the data splitting module comprises a parser program...which comprises data splitting, encryption, and decryption”); Bhattacharjee, ¶70.

Depository 210—illustrated in more detail in Figure 4 (annotated below right)—“stores cryptographic keys”; specifically, “[key] data 405” (pink below) “correspond[s]” to the cryptographic keys. Orsini, [0085]: Bhattacharjee, ¶71.



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). Bhattacharjee, ¶71.

POSAs would have further understood from these disclosures that cryptographic engine 220 and depository 210 are stand-alone components implemented on different “secure servers” (Orsini, [0066], [0079]) and that the

depository is not built into the cryptographic engine. Bhattacharjee, ¶72. In particular, Figure 4 depicts the cryptographic engine communicating with the depository using SSL (Secure Sockets Layer), a network-based security protocol. Orsini, [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). Orsini, [0081]. POSAs understood these protocols are conventionally used for communication between systems separated by different networks and/or the Internet. Bhattacharjee, ¶¶72-73 (citing EX1012, [0052], [0065]; EX1013, 3:16-29).

In some Orsini embodiments, the session master key is requested from depository 210 and used by the cryptographic engine. Orsini, [0134]. Orsini also discloses that embodiments can “stor[e] the encryption master key data separately from the data” encrypted with that master key (Orsini, [0055], [0057]), *e.g.*, as in Orsini’s FIG. 22. *See also* Orsini, [0302] (“store the session master key data in one or more separate key management table[s]”), [0303]-[0308], [0323]-[0333], FIG. 24; Bhattacharjee, ¶¶74-75.

POSAs thus 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. Orsini, [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, ¶75. Because cryptographic keys in general are stored at and received from depository 210, POSAs 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, ¶75.

Similarly, POSAs understood that in embodiments using Orsini’s “*internal* storage of session keys” ([0293]), the key-storage system from which keys are received (*e.g.*, a storage system like depository 210) would be located internal to the cryptographic engine. Bhattacharjee, ¶76; *see also infra* §VI.B.3.b (demonstrating receiving encryption keys from a storage system was also obvious over Orsini).

The above 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.

Thus, Orsini discloses “*receiving a first key from a storage system,*” as claimed. Bhattacharjee, ¶¶77-78.

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

Claim 1 recites that [1B]’s “*first key*” is involved in step [1E] to “*perform[] a cryptographic operation...to further secure the plurality of data chunks.*” As discussed above (§VI.B.3.a), Orsini’s cryptographic engine receives a session master key, which is a “*first key.*” Sections VI.B.7.a.i-VI.B.7.a.ii below demonstrate how Orsini’s session master key is used to meet [1E]. 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 [1E]. Bhattacharjee, ¶79.

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 from a “*storage system,*” POSAs had multiple reasons (discussed below) to implement Orsini to receive any of those keys from a “*storage system.*” Bhattacharjee, ¶80.

First, as discussed, Orsini provides “***session key management***” and discloses “multiple cryptographic engines” having a “single point of...key management.” Orsini, [0180], [0176]-[0179], [0293], [0381]-[0386], [0412]; Bhattacharjee, ¶81; EX1015, [0044]. POSAs would have recognized several benefits to implementing such a single point of 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 *infra*) and storing them 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, ¶81.

One benefit would have been eliminating the need (and associated complexity and cost) to install and maintain internal-key-generation and -storage mechanisms on each cryptographic engine. This would have also beneficially reduced the processing load on each cryptographic engine (by not requiring it to generate keys itself). Bhattacharjee, ¶82 (citing EX1020, 1:34-38; EX1021, [0032]; EX1055, [0006], [0009]). 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, ¶82 (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, ¶¶83-85 (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, ¶¶86-87 (citing EX1024, [0003]-[0010]; EX1025, 4:21-59, FIGs. 1-2).

Second, receiving Orsini’s encryption keys from a centralized key-storage system built for generating and storing keys would have been a simple substitution of one known element (the key-generating system within the cryptographic engine) for another (a conventional, centralized key-management system). *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 417-18 (2007); Bhattacharjee, ¶88; 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). Bhattacharjee, ¶89; EX1024, [0003]-[0010]; EX1025, 4:21-59, FIGS. 1-2.

POSAs would have reasonably expected success in each of the above implementations of Orsini's encryption key-management system. Bhattacharjee, ¶90. Orsini's "[e]ncryption key management...may be integrated into an existing infrastructure ***or any other desired location.***" Orsini, [0280]. Orsini also say keys may be stored in "a standard key store" or "custom key store." Orsini, [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, ¶90. The '609 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, 77:50-55, 78:59-79:17, 79:60-65; *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, ¶91.

Thus, Orsini also renders obvious "*receiving a first key from a storage system,*" as claimed. Bhattacharjee, ¶¶92-93.

4. [1C1] generating a plurality of data chunks based on a data set, wherein each data chunk of the plurality of data chunks comprises less than an entirety of data of the data set, and

As discussed above (§VI.B.1, 1[PRE]), an exemplary embodiment of Orsini's method is executed by a processor executing "a parser program" that performs

“encryption” “on any **data set**.” Orsini, [0278]; *see also* [0013], claims 1, 10, 20, 30; Bhattacharjee, ¶94. Although the parties dispute the definition of “*data set*” (EX1064, 6-7; EX1065, 9-10), Orsini’s use of the identical term (Orsini, [0278]) in a specification that shares disclosure and inventors with the ’609 patent meets that claim language regardless of which party’s definition is adopted.

As illustrated below, Orsini’s “data units may be distributed into one or more **shares**” (Orsini, [0116], [0296]), which meet claim 1’s “*plurality of data chunks*.” Bhattacharjee, ¶95; *see also infra* §§VI.B.4-VI.B.8 (demonstrating that Orsini’s shares meet all limitations reciting [1C1]’s “*data chunks*”). Orsini’s distribution of the data set into a plurality of **shares** (“*data chunks*”) meets both parties’ constructions of “*generating...data chunks based on a data set*” because Orsini is “splitting the data set into a plurality of data chunks” (Google; EX1064, 2-5) and “creating a plurality of data chunks based on a data set” (SFI; EX1065, 3). *See also* Orsini, [0117], [0281]-[0282], [0294]-[0296]; Bhattacharjee, ¶95.

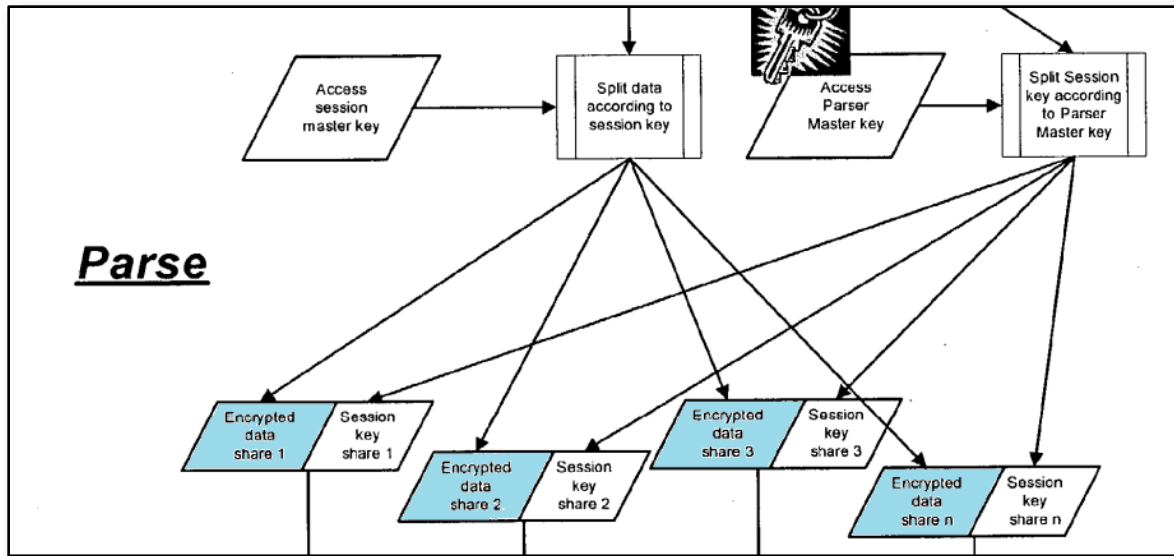


FIG. 21 (partial; annotated)

The above also illustrates that each **share** (“data chunk”) comprises a “portion[] of the original data” ([0298]) and thus “*each data chunk of the plurality of data chunks comprises less than all of the data set,*” as claimed. Bhattacharjee, ¶96.

5. [1C2] wherein the data set can be reconstructed using at least a minimum number of the plurality of chunks;

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. Orsini, [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.” Orsini, [0283]. Thus, Orsini’s “*data set can be reconstructed using at least a minimum number of the plurality of chunks*” (**shares**) under both parties’ proposed constructions, because Orsini’s “subset of the shares”

are a “predefined number [*e.g.*, three] of the lowest amount of the plurality of 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, ¶97.

6. [1D] encrypting each respective data chunk of the plurality of data chunks with a respective second key, wherein each of the respective second keys are distinct from each other;

As Figure 21 illustrates (color-annotated below), Orsini’s parser program encrypts each **share** (“*data chunk*”) with a respective **share encryption key**, which meet “*second keys [that] are each distinct from each other,*” as claimed. Orsini, [0298]-[0299], [0346]-[0352] (“multiple *distinct* secret key values: K1, K2, K3, Kn, K5” used to encrypt respective shares), [0395]-[0396]. For example, Figure 21’s “share 1 key,” “share 2 key,” *etc.*, are respective and distinct (second) encryption keys that are respective to a share (“*data chunk*”). Bhattacharjee, ¶98.

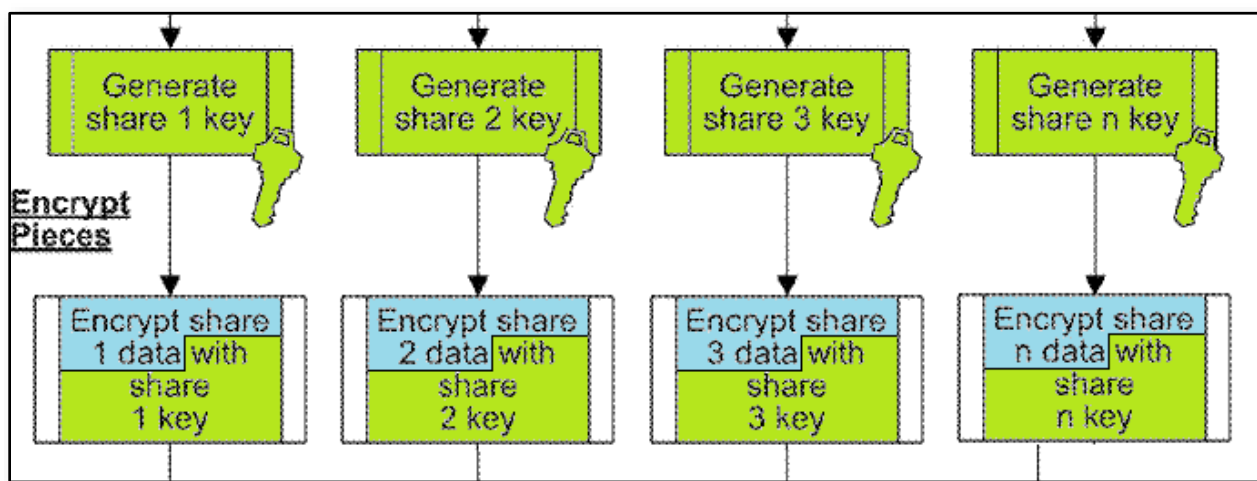


FIG. 21 (partial; color-annotations added)

7. [1E] performing a cryptographic operation based on the first key to further secure the plurality of data chunks; and

As discussed below, Orsini discloses “*performing a cryptographic operation based on the first key to further secure the plurality of data chunks,*” as claimed. Bhattacharjee, ¶99. Alternatively, implementing Orsini such that its encryption method included that step would have been obvious in view of Orsini’s disclosure, for the reasons discussed below. Bhattacharjee, ¶99.

a. Orsini discloses [1E]

Orsini discloses [1E] in two independent ways.

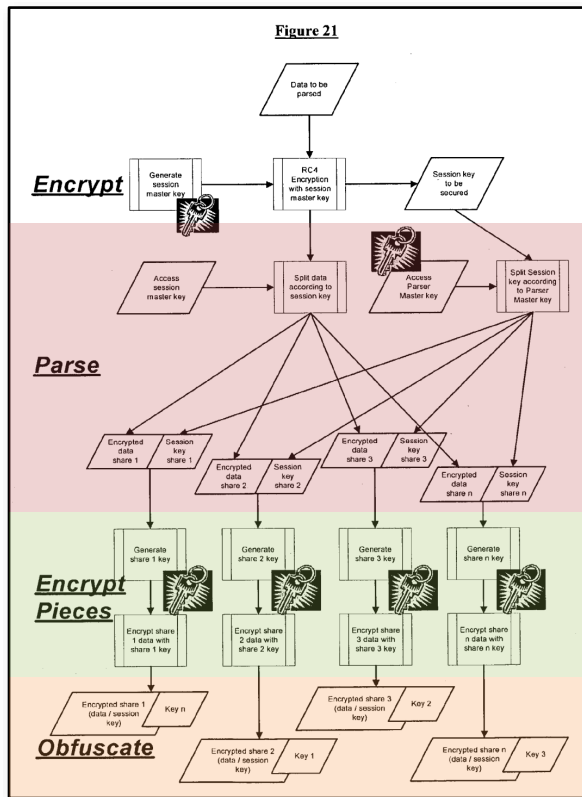
i. Performing encryption steps in different order.

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.***” Orsini, [0301]; *see also* [0279] (noting “[t]he parsing process” is “designed in a modular tiered fashion,” which reinforces the ability to change the order of encryption processing). Orsini thus embodies in which “certain steps” in Figure 21’s four encryption phases – (1) encrypt with the session master key, (2) parse, (3) encrypt with distinct share keys, and (4) obfuscate – are performed in different order. Bhattacharjee, ¶¶100-02.

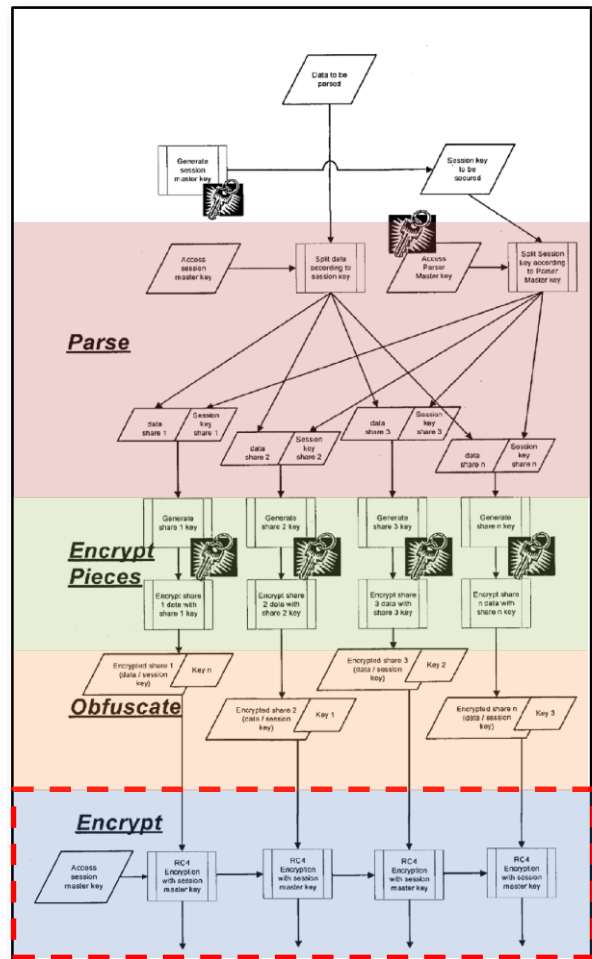
POSAs understood that one “certain step[]” capable of being reordered was the first “encrypt” step with the session master key in Orsini’s Figure 21 (discussed *supra* §VI.A) because it was well-known to reorder whether an encryption step is

performed before and/or after splitting. Bhattacharjee, ¶¶102-103 (corroborating with EX1028, [0029]; EX1053, [0026]; EX1045, FIGs. 11-13, [0109]-[0113]). Thus, one different-order embodiment includes the session master key (an exemplary “*first key*”) further encrypting data *after* it has first been parsed and encrypted with respective share keys (that is, after the data set has been split into “*data chunks*,” which are then individually encrypted), rather than before either of those steps. In this embodiment, the “*data chunks*” (Orsini’s “shares”) are “*further secure[d]*” as claimed, because they undergo a second encryption operation using the session master key, which meets the “*first key*” as explained *supra* §VI.B.3 ([1B]). Two exemplary implementations of that embodiment are conceptually illustrated below (right):

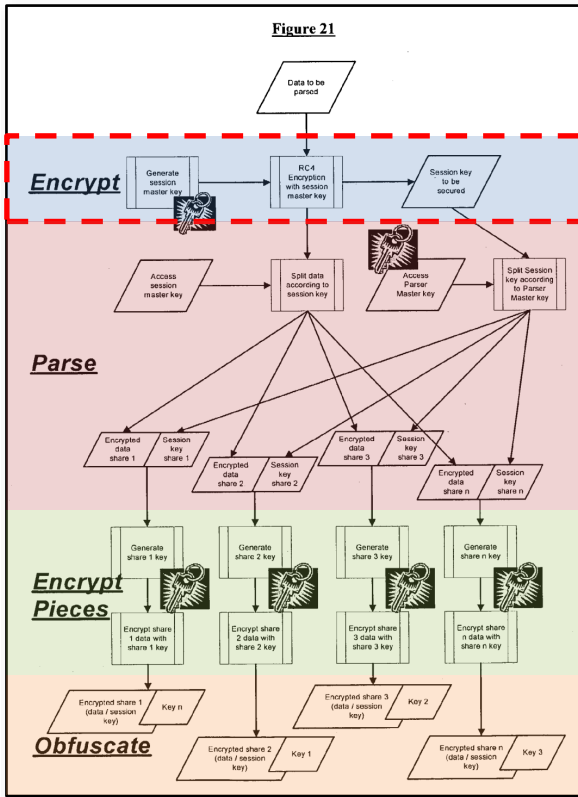
Original



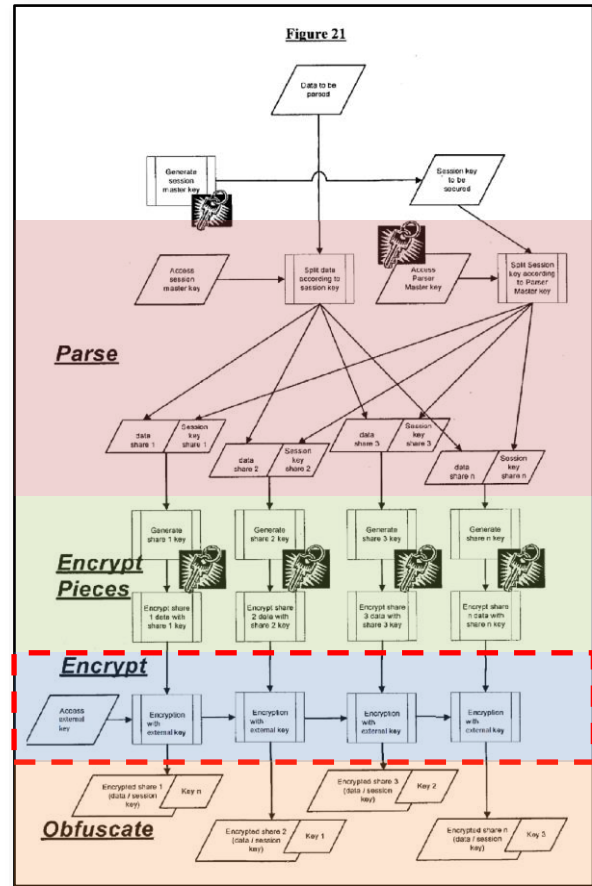
Different Order (Version 1)



Original



Different Order (Version 2)



Bhattacharjee, ¶¶104-05.

ii. Repeating the encryption steps.

Second, Orsini also teaches that Figure 21’s steps can be “*repeated multiple times.*” Orsini, [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 (claim 20’s encryption “step c),” which encrypts

portions of data, “is repeated one or more times” and “using an encryption algorithm that is different from...step a),” which “encrypt[s] a data set”), 39 (similar); Bhattacharjee, ¶106.

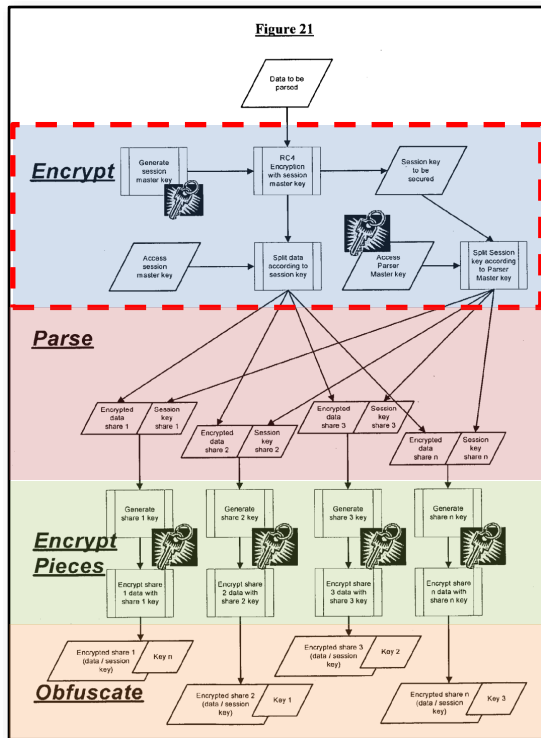
Orsini thus discloses embodiments where encrypted data shares (the encrypted “*data chunks*”) undergo an additional cryptographic operation using an encryption algorithm that differs from the algorithm used to encrypt data shares the first time (thus performing a second cryptographic operation that follows splitting the data set). Bhattacharjee, ¶107.

One different algorithm performed after encrypting shares (thus after generating the encrypted “*data chunks*”) that would have been obvious to a POSA includes using a single encryption key to further encrypt all encrypted shares with that encryption key similar to using the session master key to encrypt the entire data set before splitting it. Bhattacharjee, ¶108. 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 (“*data chunks*”), (3) encrypt the data shares with distinct share keys, (4) **encrypt each encrypted share (“*data chunk*”) with a common key (e.g., a conventional encryption key)**, and (5) obfuscate the encrypted shares. The common encryption key used to further encrypt the already-encrypted shares (the encrypted “*data chunks*”) would have been, or had reason to

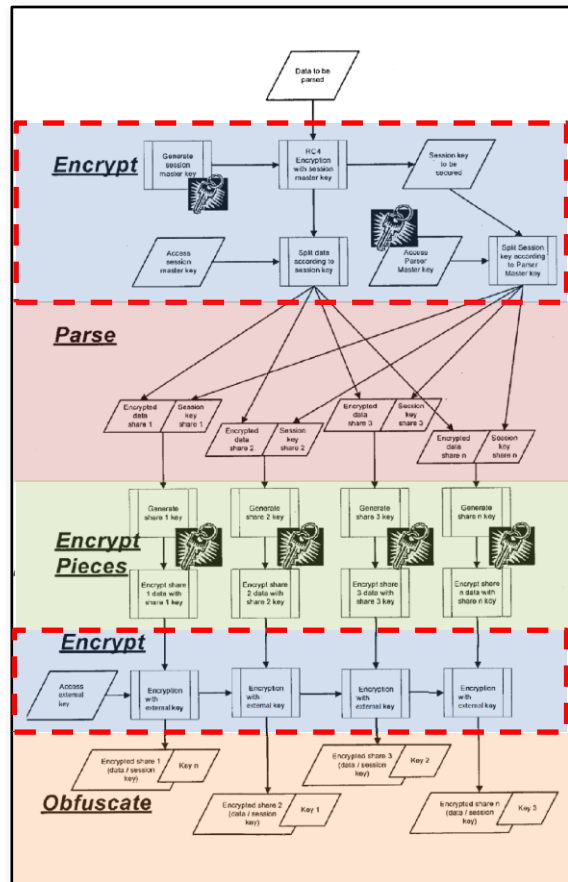
be, received from a “storage system” for the reasons discussed *supra* §VI.B.3 ([1B]), and thus meets “the first key,” as claimed. Bhattacharjee, ¶108.

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

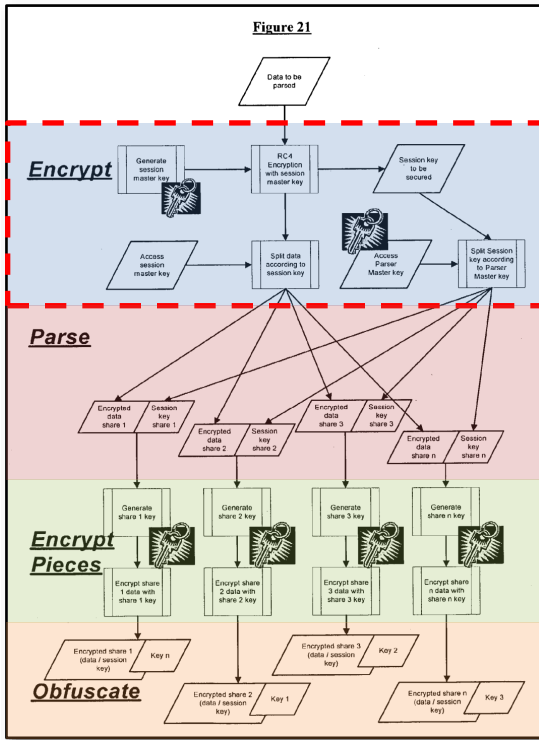
Original



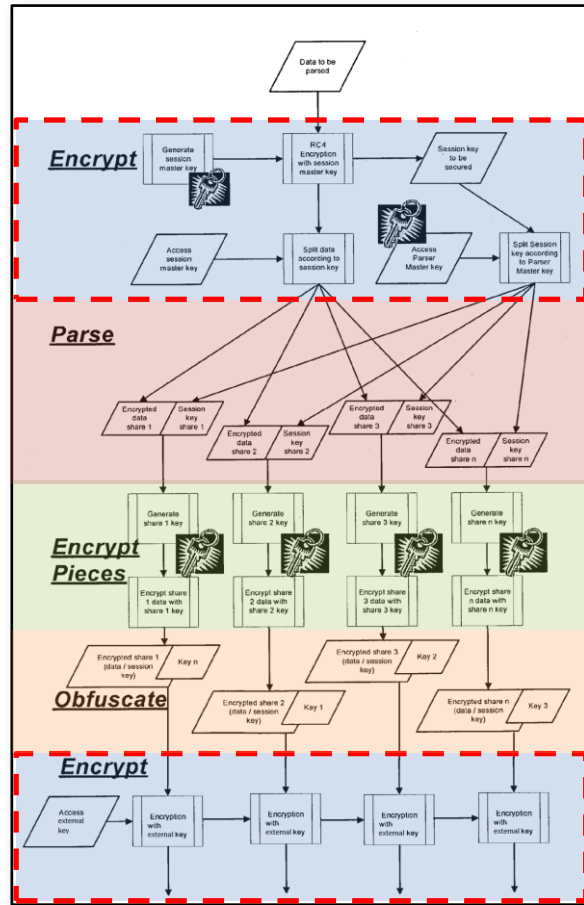
Two “Encrypt” Phases (Version 1)



Original



Two “Encrypt” Phases (Version 2)



Bhattacharjee, ¶108.

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

If SFI argues that Orsini does not explicitly disclose “performing a cryptographic operation based on the first key to further secure the plurality of data chunks,” then POSAs had multiple reasons to implement Orsini to perform that step.

i. POSAs had reason to re-encrypt data chunks with a first encryption key.

As discussed *supra* §VI.B.7.a, Orsini teaches “perform[ing]” “certain steps of the method” “in different order” or “repeated multiple times.” Orsini, [0301].

POSAs would have been motivated by these disclosures to reorder and/or repeat certain specific steps by encrypting (“*perform[] a cryptographic operation*” on) already-encrypted data chunks with the same encryption key, and would have understood that doing so would have provided additional security and efficiency benefits. Bhattacharjee, ¶¶109-11; *see supra* §VI.B.3 (demonstrating Orsini discloses or alternatively renders obvious receiving its encryption keys from a “*storage system*”); *infra* §§VI.B.7.b.ii-VI.B.7.b.iii (discussing benefits of further cryptographic operations on data chunks); §VI.B.7.b.iv *infra* (discussing known benefits of encrypting encryption keys, which SFI contends meets the limitation).

Re-encrypting data chunks with a common 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, ¶¶112-13 (citing EX1028, [0090]; EX1053, [0026]; EX1045, [0109]-[0113]).

ii. POSAs had reason to re-encrypt data chunks with a 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 those embodiments would have been to encrypt shares (the “*data chunks*”) with a group-level encryption key, thereby restricting user access to shares based on membership in authorized group(s). *KSR*, 550 U.S. at 417 (obvious to use known technique to improve similar devices); Bhattacharjee, ¶114 (explaining that group-level encryption was conventional and citing corroborating evidence 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 a storage system for the same reasons discussed above (§VI.B.3.b, [1B]) that POSAs had reason to receive any other encryption keys from a storage system. Bhattacharjee, ¶¶114-15.

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

POSAs also had reason to add additional security to Orsini’s encrypted shares (encrypted “*data chunks*”) by cryptographically reordering/shuffling the data *within*

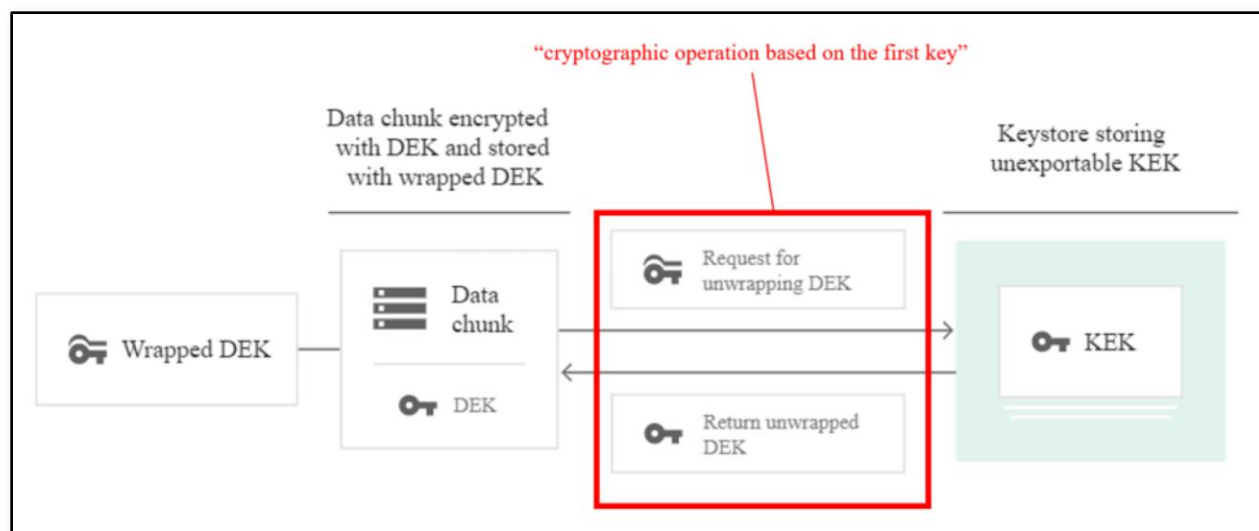
each “*data chunk*,” which was a conventional cryptographic operation to secure data. EX1030, [0002]. This is expressly taught (or alternatively rendered obvious) by Orsini’s disclosure that the “distribution” of the dataset’s data “could also involve a shuffling of the order of the units [of the data] in the shares.” Orsini, [0281]. POSAs understood that the benefits from shuffling data would also accrue if the shuffling were performed after encrypting each share with a respective, distinct encryption key (that is, shuffling the data within each encrypted “*data chunk*”). And, as noted above (§VI.B.7.a.i), Orsini explicitly teaches performing encryption steps in different order. 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 encrypted “*data chunk*” (each encrypted share). In a conventional technique, the reordering pattern for portions (*e.g.*, bits/bytes) of data would be defined using an encryption key. Bhattacharjee, ¶¶116-117; 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 shares (“*data chunks*”) with a cryptographic operation based on one of the “*first keys*” received from a “*storage system*” discussed above (§VI.B.3), as claimed, by using the received first key to define the reordering pattern within each encrypted “*data chunk*.” Bhattacharjee, ¶¶118-19.

iv. POSAs had reason to encrypt [1D]’s “*respective second keys*” with a key-encryption key.

Orsini discloses using a so-called “public key” “*to encrypt the session key* that will be used to encrypt the document or message.” Orsini, [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, ¶120 (citing EX1035, EX1036).

As discussed below (§VI.B.8, [1F]), Orsini stores the share encryption keys (with encrypted data chunks) on storage systems. However, POSAs knew, in general, “cryptographic keys should never occur in clear form outside the cryptographic device.” EX1035, 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, alternative or in addition to encrypting session keys with a KEK, POSAs had reason to encrypt [1D]’s keys with a KEK, to avoid a well-known security risk in storing those keys in clear (unencrypted) form on storage devices (especially those accessed across the public Internet). Bhattacharjee, ¶121 (explaining the use of KEKs was conventional and citing as corroboration 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 [1E] in related district court litigation, encrypting [1D]’s keys with a KEK meets [1E]. See EX1009, 8-9 (arguing “DEKs are encrypted with (wrapped by) a key encryption key (KEK)”). POSAs had reason to receive the KEK retrieved from a “storage system” (thereby meeting the claimed “*first key*”) for the same reasons POSAs had to receive Orsini’s session master key from a storage system, as discussed above (§VI.B.3.b, [1B]). Bhattacharjee, ¶122.



* * *

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 cryptographic 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 [cryptographic method] steps may be desired” ([0230]). Bhattacharjee, ¶123.

8. [1F] storing, in a memory coupled to the processor, at least one data chunk of the plurality of data chunks with data indicative of at least one of the distinct encryption keys on at least one storage device.

Though unartfully written, POSAs understand [1F] to require, *inter alia*, storing data “on at least one storage device” in “memory” (e.g., a disk drive of a storage device) that is “coupled to the processor.” The ’609 patent describes a component “***coupled to*** [a] storage area [] ***using any suitable communication interface*** [] (e.g., USB, serial, parallel, Bluetooth, IR, IEEE 1394, Ethernet, ***or any other suitable communications interface***).” EX1001, 66:49-55. Thus, POSAs understood the patent uses “coupled to” broadly to include a connection using any communication interface that allows two components to exchange information. POSAs would, therefore, understand [1F]’s “coupled” to be met if the processor can communicate information (specifically “at least one data chunk”) to the memory. Bhattacharjee, ¶124. Orsini meets this as discussed further below.

With respect to data stored with [1D]’s respective encryption keys (the “second keys”), Orsini’s disclosure is substantively identical to the ’609 patent’s disclosure. Thus, Orsini discloses [1F] to the extent the ’609 patent does. Bhattacharjee, ¶125. For example, SFI cites the EX1001 at 31:34-40, 48:28-38, and

74:52-56 as intrinsic evidence that allegedly supports its construction of the “data indicative” term. See EX1065, 4. EX1001 at 31:34-40 and 48:28-38 are identically disclosed in Orsini at [0162] and [0247], respectively. EX1001 at 74:52-56 discloses using a “hash function” with “random or pseudo-random value” to “indicate to which share to append” a “respective bit” of data. Orsini similarly discloses “perform[ing] hashing” and “hash comparisons” (Orsini, [0134]) “partition[ing] the data” “based on...random” values (Orsini, [0280]-[0281]). Thus, Orsini discloses [1F] to the same extent the ’609 patent does. Bhattacharjee, ¶125.

As Figure 21 illustrates (partially shown and color-annotated below), Orsini stores **encrypted shares** (encrypted “*data chunks*” §VI.B.4) with a share encryption key (“*respective second keys*”/“*distinct encryption keys*,” §VI.B.6). 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 also EX1009, 10-11 (alleging [1F] is met by “stor[ing] DEKs”—the keys alleged to meet [1D]—“with the data chunks”).

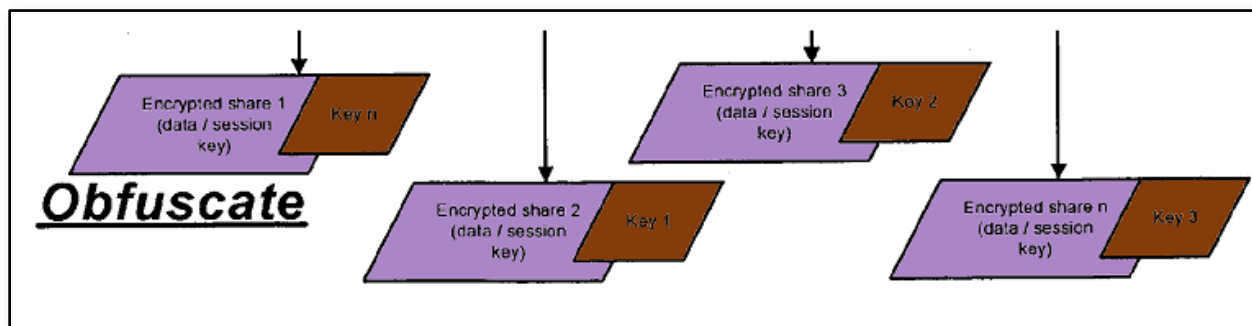


FIG. 21 (partial; color-annotated)

The share encryption keys used to encrypt shares (“*data chunks*”) are stored “in different locations from the encrypted data portions or shares” (e.g., “Share 1 gets Key 4” etc.). Orsini, [0299]. These different locations store the encrypted **share** (encrypted “*data chunk*”) and **share encryption key** (“*distinct encryption key*”) “on one or more data depositories or **storage devices**” (“*on at least one storage device*”). Orsini, [0379]; *see also* [0018], [0111]-[0112], FIGS., 7, 21, Claims 10, 13-16. These storage devices and their memory are coupled to the processor performing Orsini’s methods (*see* §VI.B.2) because the processor communicates the data chunks to the memories in these storage device(s). *See also* EX1009, 9 (alleging [1F] is met by storing data chunks on “storage devices...throughout Google’s storage infrastructure”); Bhattacharjee, ¶126.

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. Orsini, [0378]. POSAs understood that AES (and other conventional encryption algorithms) supported multiple so-called encryption modes. Bhattacharjee, ¶127 (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 construction of “*data indicative of...[a] distinct encryption key.*” Bhattacharjee, ¶127.

POSAs also understood that it was conventional to support variable-sized encryption keys, to provide additional security. Bhattacharjee, ¶128 (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 construction. EX1064, 5-6. Bhattacharjee, ¶128.

Thus, Orsini discloses and renders obvious [1F] under both parties’ proposed constructions. Bhattacharjee, ¶129.

As discussed above (§§VI.B.3-VI.B.8), Orsini performs limitations [1B]-[1F] 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, ¶130.

C. Claim 2: The method of claim 1, further executing the code by the processor to perform: storing with each data chunk of the plurality of data chunks data indicative of at least one of the distinct encryption keys.

As discussed above (§VI.B.8, [1F]), Orsini’s processor executes code to store with each share (“*data chunk*”) a share encryption key (“*distinct encryption key*”),

which meets the limitation added by claim 2 under SFI's claim constructions (EX1065, 4) and infringement contentions (EX1009, 11-13). Bhattacharjee, ¶131.

Under Google's constructions (EX1064, 5-6), Orsini renders obvious the limitation added by claim 2, for the reasons discussed *supra* §VI.B.8 ([1F]) that Orsini renders obvious storing data chunks with metadata about the “*distinct encryption keys*” (*e.g.*, the title key length and/or associated encryption mode), which is “*data indicative*” of those keys under Google's constructions. Bhattacharjee, ¶132.

D. Claim 3: The method of claim 1, further executing the code by the processor to perform: storing with each data chunk of the plurality of data chunks data indicative of a distinct encryption key used to encrypt that data chunk.

As discussed above (§VI.B.8, [1F]), Orsini discloses embodiments that store each share (“*data chunk*”) with the share encryption key (a “*distinct encryption key*”) used to encrypt ***another*** share. However, Orsini discloses that this is ***optional***. *E.g.*, Orsini, [0406] (“Share Encryption Keys ***may*** be stored in different shares....”). In fact, Orsini has claims reciting storing each portion of data with its respective encryption key. For example, Orsini's claim 1 recites a step “c) encrypting one or more of the portions of data” and step “d) storing the encrypted portion...on one or more data depositories” and Orsini's claim 7 recites that “the encryption of step c) provides an encryption key, and wherein the encryption key is stored in step

d) **together with the data encrypted using said encryption key in step c).**”

Bhattacharjee, ¶133; *see also* Orsini claims 16, 26.

Thus, Orsini explicitly discloses in its claims (which are prior art to the ’609 patent) storing each share with the share encryption key (“*distinct encryption key*”) that was used to encrypt *that* share, rather than with another share’s key. Bhattacharjee, ¶134; *KSR*, 550 U.S. at 418.

Beyond Orsini’s disclosure, POSAs had independent reason to store each share (“*data chunk*”) with the share encryption key (“*distinct encryption key*”) used to encrypt that share.

First, as discussed above, Orsini’s claims 7, 16, and 23 (at least) suggest storing each share (“*data chunk*”) with its respective share encryption key (“*distinct encryption key*”). Bhattacharjee, ¶¶135-36.

Second, storing each share with its respective share encryption key would have beneficially reduced the processing required to reconstruct the data set (by not obfuscating share-key combinations) and facilitated decryption of, and access to, individual shares. Bhattacharjee, ¶137 (citing EX1006, [0007], [0048]); *KSR*, 550 U.S. at 418. This would have been a conventional tradeoff in encryption-method design, between, on the one hand, convenience and efficiency – by storing shares with indexes or encrypted versions of their respective keys, which would allow, *e.g.*, easier decryption of individual shares – and, on the other hand, additional security –

by storing shares with encryption keys for other shares. Bhattacharjee, ¶138 (citing EX1061, [0059]; EX1006, [0042], [0048], FIGS. 3A-3B).

Third, it would have been a simple substitution of a known and conventional technique (storing encrypted data with its encryption key) for another, to obtain the predictable result that each encrypted share would be stored with its share encryption key. *KSR*, 550 U.S. at 417; Bhattacharjee, ¶139; EX1006, [0042], [0048], FIGS. 3A-3B.

As discussed above (§VI.B.8, [1F]), under SFI's constructions, a distinct encryption key is "*data indicative of*" that key. EX1065, 3-4; *see also* EX1009, 13-14 ("each DEK stored with a data chunk is used to encrypt that particular data chunk"); Bhattacharjee, ¶140.

As discussed above (§VI.B.8, [1F]), POSAs alternatively had reason to store each share with metadata about a key (*e.g.*, its length and/or associated encryption-mode), which meets Google's proposed constructions. EX1064, 5-7. When doing so, it would have also been conventional to store that metadata with its corresponding share ("*data chunk*"), as opposed to another share, because, as discussed above, it would have been a conventional implementation and a beneficial implementation, depending on how the system designer weighed the well-understood tradeoff between data security and convenience, discussed above. Bhattacharjee, ¶141.

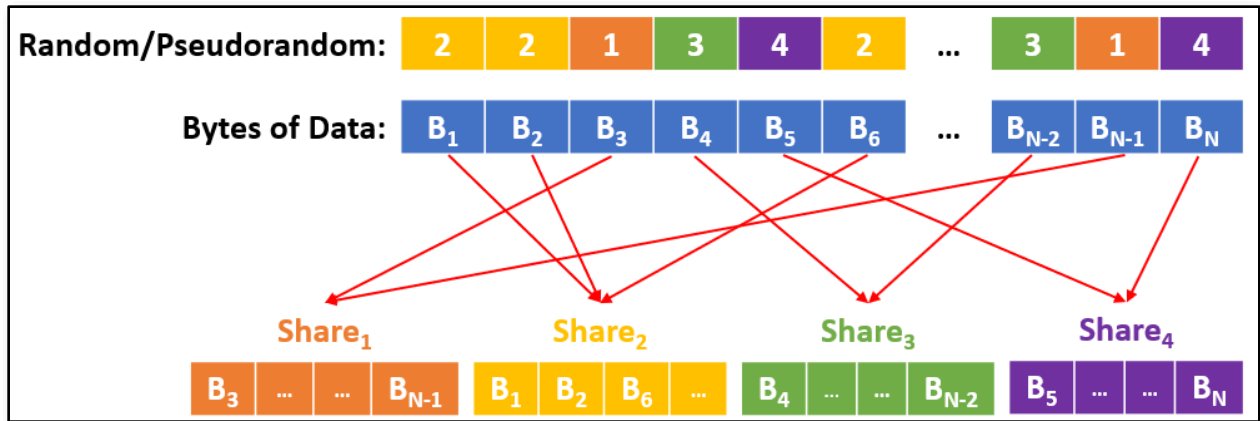
POSAs would have reasonably expected success because as noted above Orsini claims it and POSAs independently understood it would have required conventional computer-programming skills to distribute keys and/or metadata to respective shares and store the keys and/or metadata with the share. Bhattacharjee, ¶142.

Thus, Orsini discloses and renders obvious claim 3 under both parties' proposed constructions. Bhattacharjee, ¶143.

E. Claim 4: [1C1]'s generating...comprises using a random technique or pseudorandom technique.

As discussed above (§VI.B.4, [1C1]), Orsini's exemplary method in Figure 21 distributes the data set to generate shares ("*data chunks*") based on the session master key. Orsini, [0087], [0280], [0393]-[0394]. Orsini discloses several different ways to perform that step, including random or pseudorandom techniques. Bhattacharjee, ¶144.

For example, Orsini discloses a "cryptographic split process" that generates, *e.g.*, "4 shares" ("*data chunks*") from the data set based on a "***random*** distribution" of numbers "corresponding to the four shares," which is a random technique for "*generating the plurality of data chunks*," as claimed. Orsini, [0282]; *see also* [0105], [0116], [0124], [0281], [0284], Claim 54; Bhattacharjee, ¶¶145, 147. That technique is illustrated below:



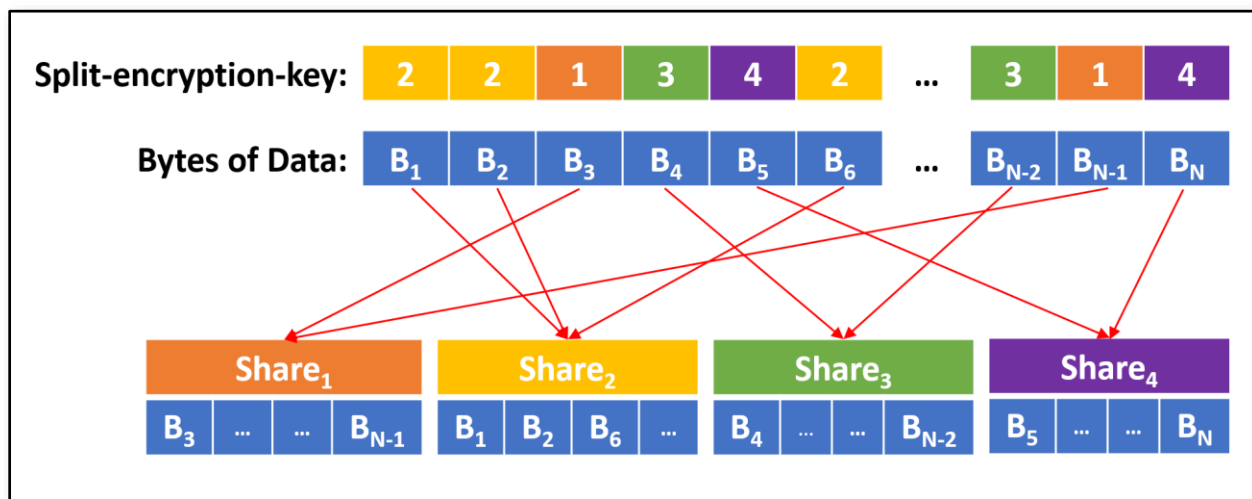
Additionally, Orsini discloses that data can be “parsed into ‘n’ segments, or shares, according to the value of K5,” which is a “secret key value[.]” Orsini, [0346]-[0348]. This “results in ‘n’ *pseudorandom* shares of the original encrypted data” being generated ([0348]), and thus is a pseudorandom technique, as claimed. Bhattacharjee, ¶¶146-47.

F. Claim 5: [1C1]’s generating...comprises a substantially random distribution of a subset of the data set.

Orsini meets claim 5 because the random or pseudorandom techniques discussed for claim 4 result in a substantially random distribution of a subset of the data set in each share, as discussed *supra* §VI.E. Bhattacharjee, ¶148.

G. Claim 6: [1C1]’s generating...comprises: generating a split encryption key based on the first key; and distributing the data set based on the split encryption key.⁷

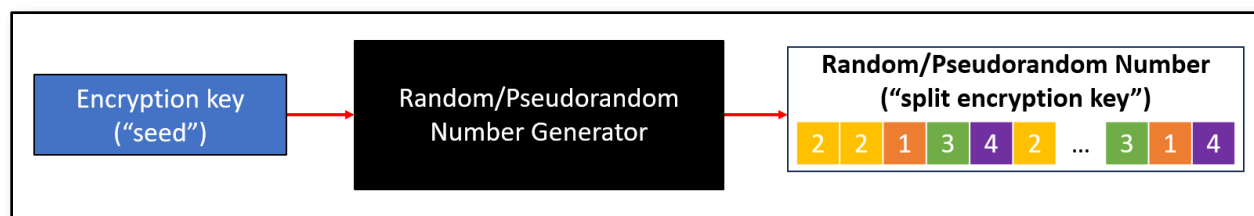
As discussed above (§VI.E, claim 4), Orsini discloses generating shares (“*data chunks*”) by distributing the data set into shares based on a random or pseudorandom technique. For example, Orsini discloses “*a key* would be obtained to create a sequence of 23 random numbers...each with a value between 1 and 4 corresponding to the four shares.” Orsini, [0282]. As illustrated below, that number sequence is a “*split encryption key*” because it is the key used in the algorithm for distributing data into shares. Bhattacharjee, ¶149; see EX1001, 77:35-38 (“split encryption key” is a key used in the “split algorithm” that “split[s data] into a predefined number of shares”).



⁷ Claims 6-10 as addressed herein account for SFI’s granted certificate of correction.

See EX1002, 12-16; §IV.C *supra*.

As noted above, Orsini says the random/pseudorandom number (“*split encryption key*”) is itself created using a key. Orsini, [0282]. This is consistent with a conventional random/pseudorandom number generator (illustrated below) that inputs an encryption key as a so-called “seed.” Bhattacharjee, ¶¶150-52 (citing EX1040, [0030] (encryption key as “seed” for “pseudo random number generator”); EX1041, [0024] (“Typically...the so-called seed, will be the...session key”), [0026]-[0027] (“output sequence” of number generator is a “key sequence”)).



As discussed above (§VI.B.4, [1C1]), Orsini generates shares (“*data chunks*”) based on the session master key (the claimed “*first key*”—*supra* §VI.B.3). Orsini, [0296], FIG. 21. Thus, in Orsini’s embodiments that use a random/pseudorandom technique to generate shares (“*data chunks*”), the session master key is the key used to generate the random/pseudorandom number (i.e., the “split encryption key”). Bhattacharjee, ¶152. Alternatively, POSAs had reason to generate that key using the session master key (or another conventional encryption 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, ¶153.

Thus, Orsini discloses, or alternatively renders obvious, using a “*split encryption key*” that is based on the session master key or another conventional encryption key implemented as a “*first key*,” and distributing the data set into shares (“*data chunks*”) based on that split encryption key, as claimed. Bhattacharjee, ¶154.

H. Claim 7: The method of claim 1, further executing the code by the processor to perform: encrypting each of the chunks of the plurality of chunks with the first key.

As discussed above (§VI.B.7, [1E]), Orsini discloses and renders obvious encrypting all shares (“*data chunks*”) with the same “*first key*,” *e.g.*, Orsini’s session master key (or another conventional encryption key), as claimed. Bhattacharjee, ¶155.

For example, as discussed above, Orsini discloses (§VI.B.7.a.i) and POSAs had reason to re-order Figure 21 by encrypting “*data chunks*” (rather than the original data set) with the session master key (§VI.B.7.b.iii) and/or a group-level key (§VI.B.7.ii). As another example discussed above, Orsini discloses (§VI.B.7.a.ii) and POSAs had reason (§VI.B.7.b.i) to repeat the splitting of data with the session master key. Each of those embodiments encrypts the shares with the session master key (“*first key*”), as claimed. *See also* Orsini, [0119] (“further secur[ing] data chunks” with a public key); Bhattacharjee, ¶156.

Finally, if SFI argues encrypting [1D]’s “*distinct encryption keys*” with a key-encrypting key (KEK) meets claim 7, it would have been obvious to do so using a KEK received from a storage system, for the reasons discussed above (§VI.B.7.b.iv). Bhattacharjee, ¶157.

I. Claim 8: The method of claim 7, wherein the data set is recoverable using at least a minimum number of the plurality of data chunks and the first key.

Orsini meets claim 8 for the reasons discussed above for [1C2] (§VI.B.5) and claim 7 (§VI.H), because “[t]o restore the original data format, the steps are reversed.” Orsini, [0300]; Bhattacharjee, ¶158.

**J. Claim 9: Claim 1’s first key is an AES key;
Claim 10: Claim 1’s first key is an DES key.**

Orsini discloses that “a wide variety of encryption methods or algorithms...are suitable for use in the present invention, including...Triple **DES** and **AES**.” Orsini, [0378]. Thus, Orsini discloses embodiments with its encryption keys, including the encryption key that meets [1B]’s “*first key*” (*e.g.*, Orsini’s session master key or another conventional encryption key), implemented as DES or AES keys, as claimed. Bhattacharjee, ¶159.

Implementing Orsini’s encryption keys as AES or DES keys would have also been obvious. Orsini’s session master key, for example, both encrypts and decrypts data, and thus is a so-called “symmetric” encryption key. Orsini, [0168]; EX1001, 32:57-61 (noting that “symmetric key cryptographic systems” are “systems where

the sender and receiver of a message share a single common key that is used to encrypt and decrypt a message”). For symmetric encryption keys, POSAs had reason to use an AES- or DES-compliant key because AES and DES were well-known, conventional (and government-approved) symmetric-key encryption standards published years before the earliest-potential priority date. Thus, using an AES or DES key for Orsini’s “*first key*” (whether the session master key or another conventional encryption key) would have been beneficial because those were well-tested and well-understood encryption techniques in the prior art. Bhattacharjee, ¶¶161-62 (citing as corroboration EX1057, [0006]; EX1058, 1:12-48, 4:52-54).

Therefore, Orsini anticipates or alternatively renders obvious claims 9 and 10.

VII. GROUND 2: CLAIMS 1-10 ARE UNPATENTABLE OVER FOSTER

As discussed below, claims 1-3, 7-8 are anticipated by or alternatively obvious over Foster; claims 4-6, 9-10 are obvious over Foster.

A. Foster (EX1006)

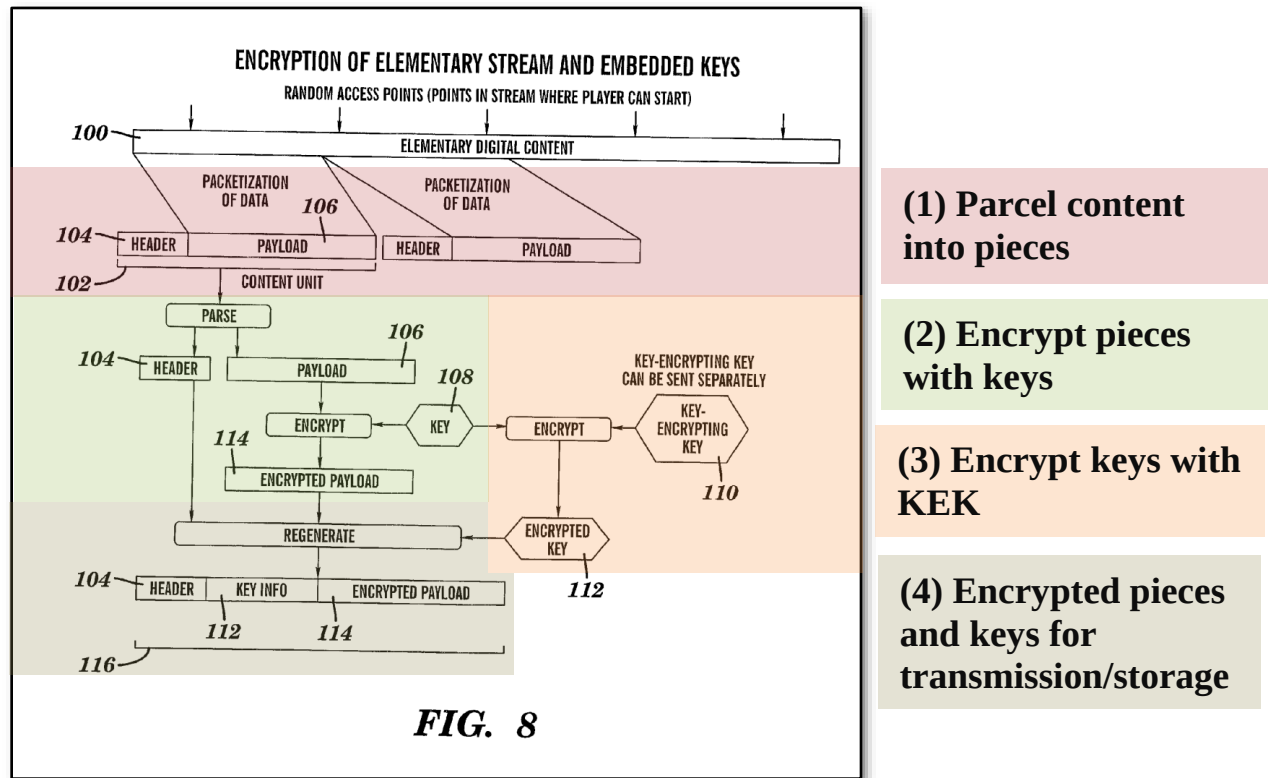
Foster discloses methods for securing digital content, like a digital movie, using encryption techniques that “attach[] a title [encryption] key to encrypted content for synchronized transmission to, or storage by, a recipient.” Foster, Abstract, [0002]. Foster’s digital content includes “*any* data.” Foster, [0037]; Bhattacharjee, ¶¶162-64.

Foster teaches that prior-art encryption methods used “multiple keys” to “provide[] enhanced security by preventing an entire content stream from being accessed with a single key.” Foster, [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, and also created challenges in “provid[ing] random access” to portions of content. Foster, [0006]; Bhattacharjee, ¶165.

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 packets, and (5) transmitting the encrypted content packets to receivers (*e.g.*, devices in a consumer's home). Foster, [0040]-[0041], [0064], FIG. 8; Bhattacharjee, ¶166.

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



Bhattacharjee, ¶167.

First, “content stream 100” “is parceled into content units 102.” Foster, [0064]. Each content unit comprises a “payload” portion of the original data set (also referred to as “content packet 106”) and a “header 104,” which “includes

information helpful in identifying the content” (*e.g.*, “type of content”). Foster, [0064], FIG. 8; Bhattacharjee, ¶168.

Second, each payload portion is “encrypted with [a] title key 108” to “yield encrypted content packet 114.” Foster, [0064]. Rather than using a common title key for all portions, “each content packet could be encrypted with a *different* title key.” Foster, [0065]; Bhattacharjee, ¶169.

Third, each title key is encrypted with a common “key encrypting key 110” (*i.e.*, a KEK, *see* §VI.B.7.b.iv above) that is received from an external storage system. Each “encrypted title key 112” is then “attached to the encrypted content as a header extension.” Foster, [0048], [0064], [0066], FIGS. 8-10; Bhattacharjee, ¶170.

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). Foster, [0040]-[0041], [0064]; Bhattacharjee, ¶171.

Foster’s Figure 16 (color-annotated below) depicts an exemplary system in which its encryption technique can be employed. Foster, [0030], [0075]. Foster’s KEK discussed above is stored in “[d]atabase 426” (beige). Foster, [0046]; *infra* §VII.B.3. The system also includes a “source 428” (green), which can be a “content owner...such as a movie studio.” Foster, [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.” Foster, [0039]-[0041], [0078]; Bhattacharjee, ¶¶172-73.

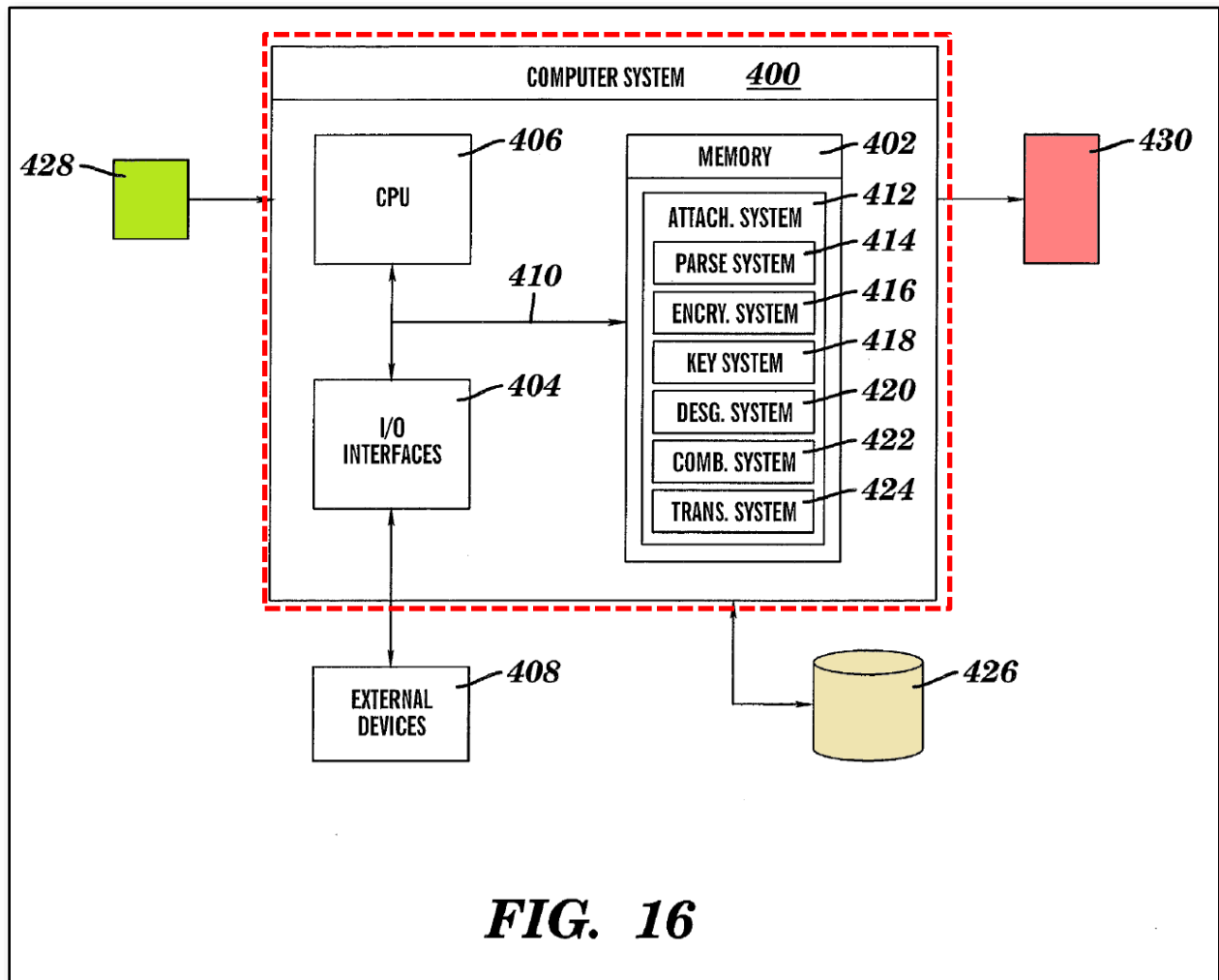


Figure 16’s system includes “computer system 400” (with a computer in the red-dashed box above) which is “any system capable of processing content units of

an elementary content stream received from source 428.” Foster, [0078]. Because this system 400 is interposed between source 428 and recipient 430, POSAs would have understood Foster to disclose that system 400 can perform functions of the “Content Service Provider” that acts 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, ¶173.

B. Claim 1:

1. 1[PRE]

Foster performs a “*method for securing data*,” as claimed.

Foster’s Figure 8 (color-annotated below) illustrates 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, ¶¶174-75.

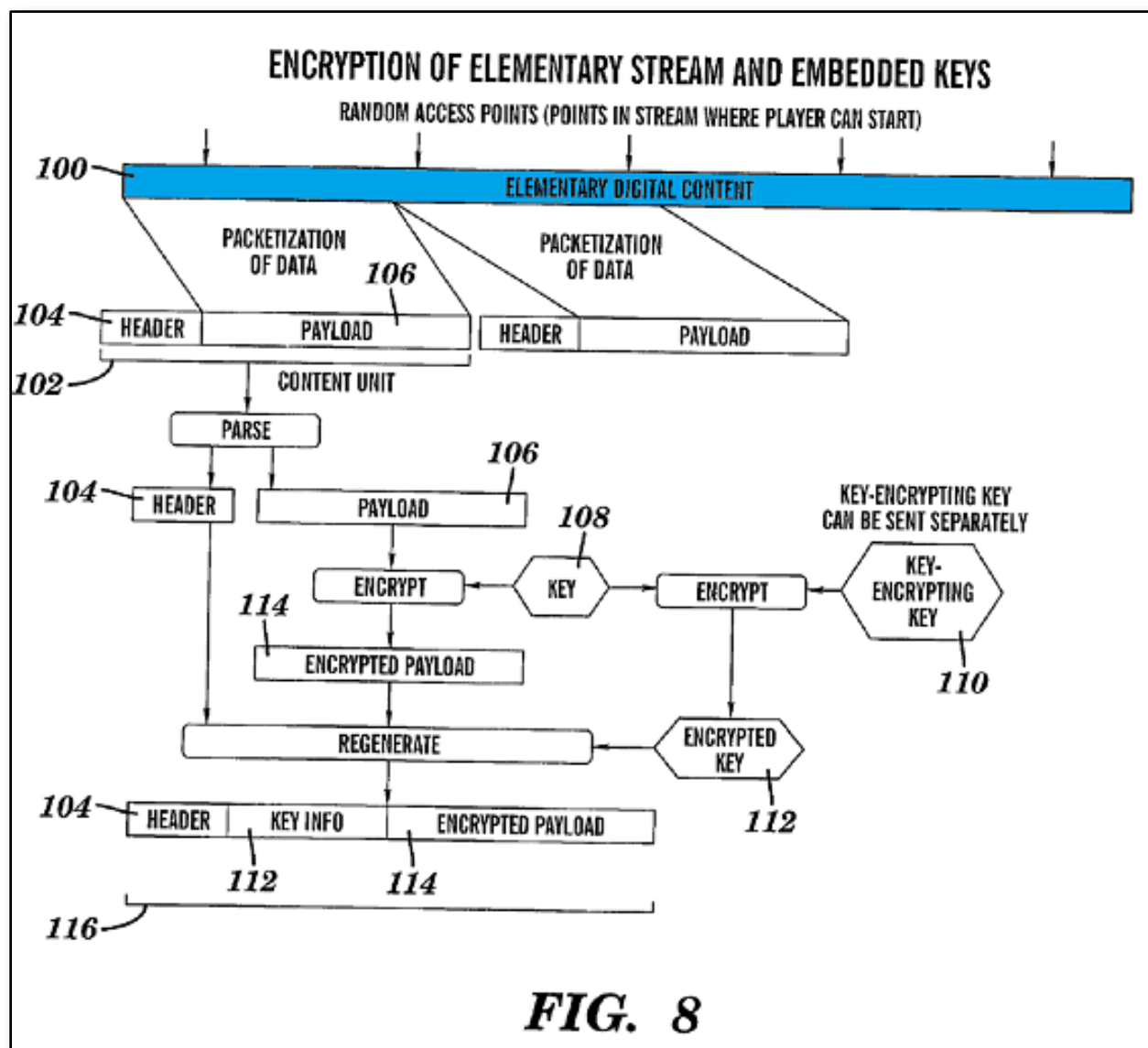
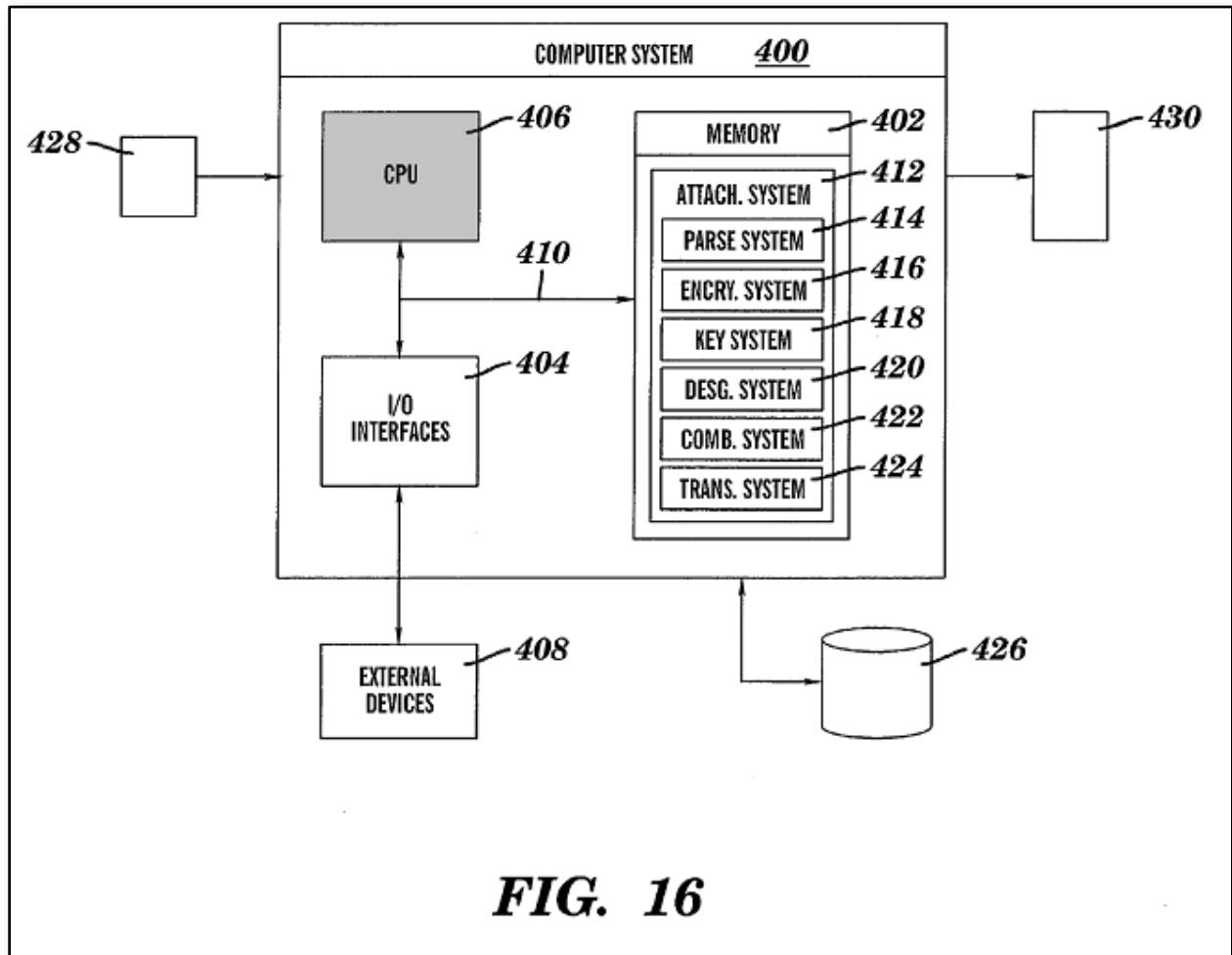


FIG. 8

2. [1A]

Foster “*execut[es] code by a processor to perform*” Foster’s method, as claimed.

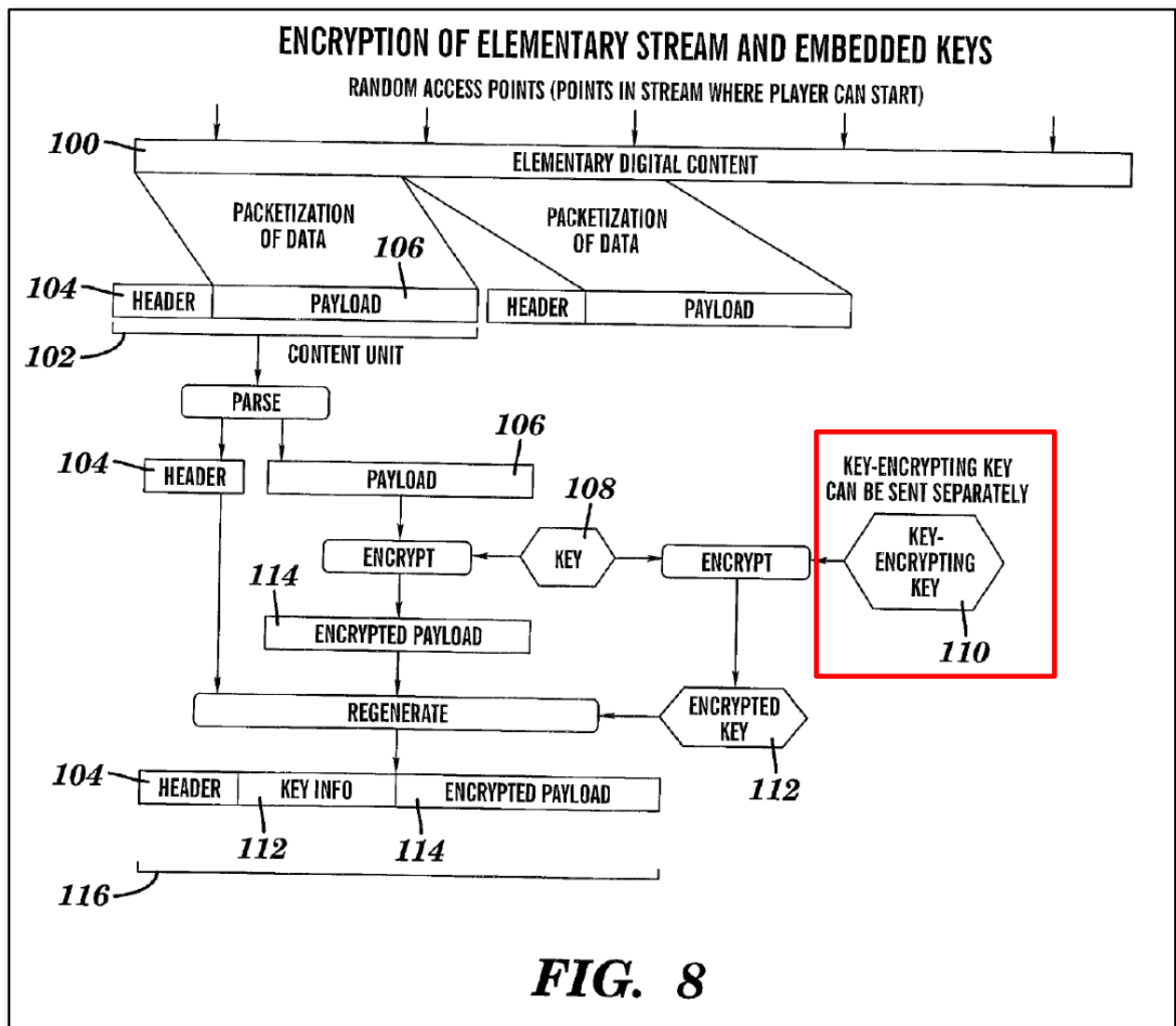
Foster’s methods are implemented by executing “a computer program,” which comprises “program *code*,” using a computer system’s CPU (central processing unit), which is a “*processor*.” Foster, [0074]-[0075], FIG. 16 (annotated below), [0012] (“*execut[ing]*” “program code”); Bhattacharjee, ¶¶176-77.



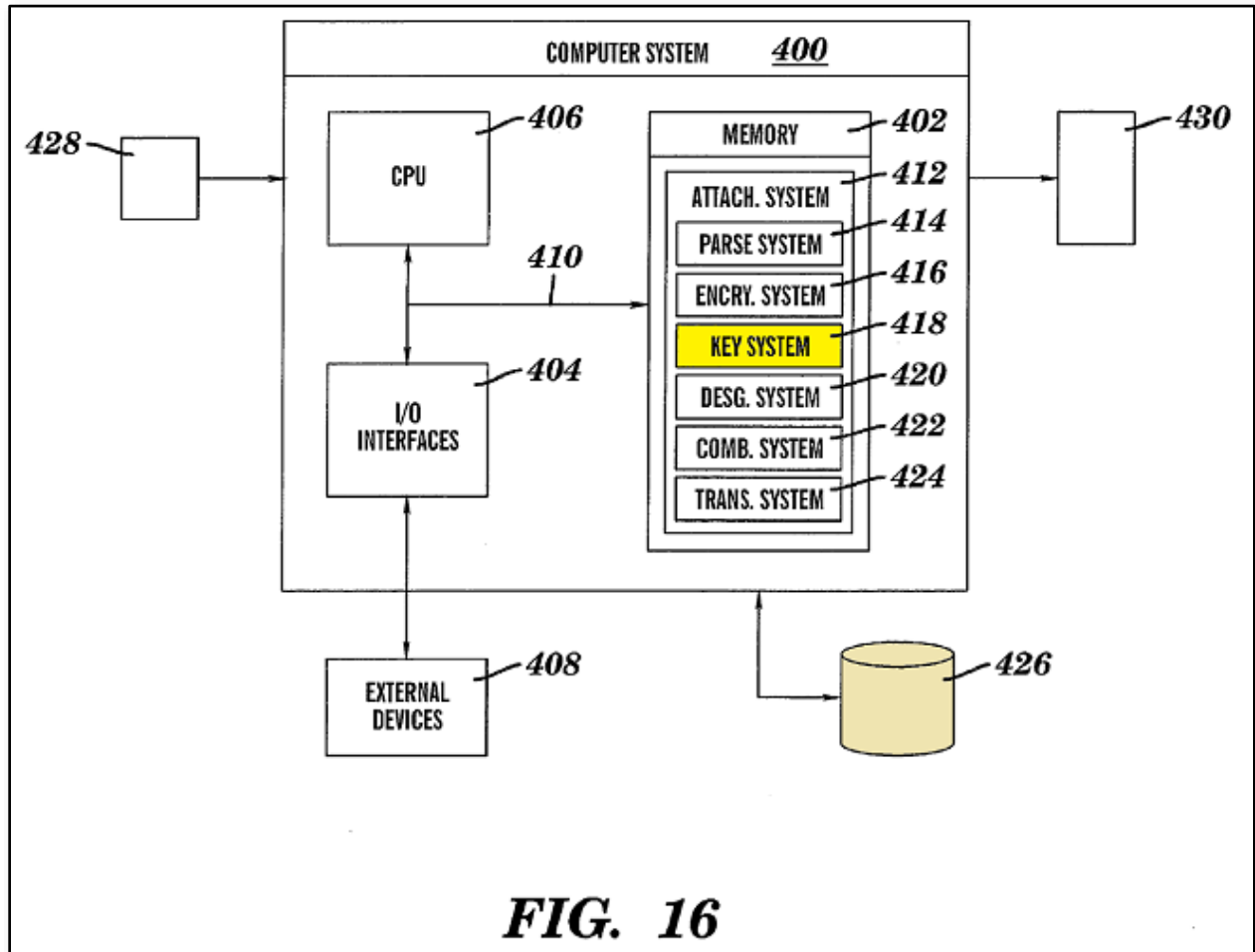
3. [1B]

Foster “receiv[es] a first key from a storage system,” as claimed.

As discussed in more detail below (§VII.B.7, [1E]), Foster’s encryption method includes a step of encrypting distinct title keys with a key encrypting key (KEK), which is a “first key” as explained below. Bhattacharjee, ¶¶178-79. This step is illustrated in Foster’s Figure 8 (annotated below):



As shown in Foster's Figure 16 (annotated below), Foster's encryption method uses a "key system 418." Foster, [0078]-[0079], FIG. 16. Key system 418 "use[s] the key encrypting key...to encrypt the title key[s]." Foster, [0079], [0046]; Bhattacharjee, ¶180.



"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, ¶¶181-82.

When encrypting content, computer system 400 receives the KEK (“*first key*”) from database 426, and therefore Foster “*receiv[es] a first key from a storage system,*” as claimed. Bhattacharjee, ¶183.

4. [1C1]

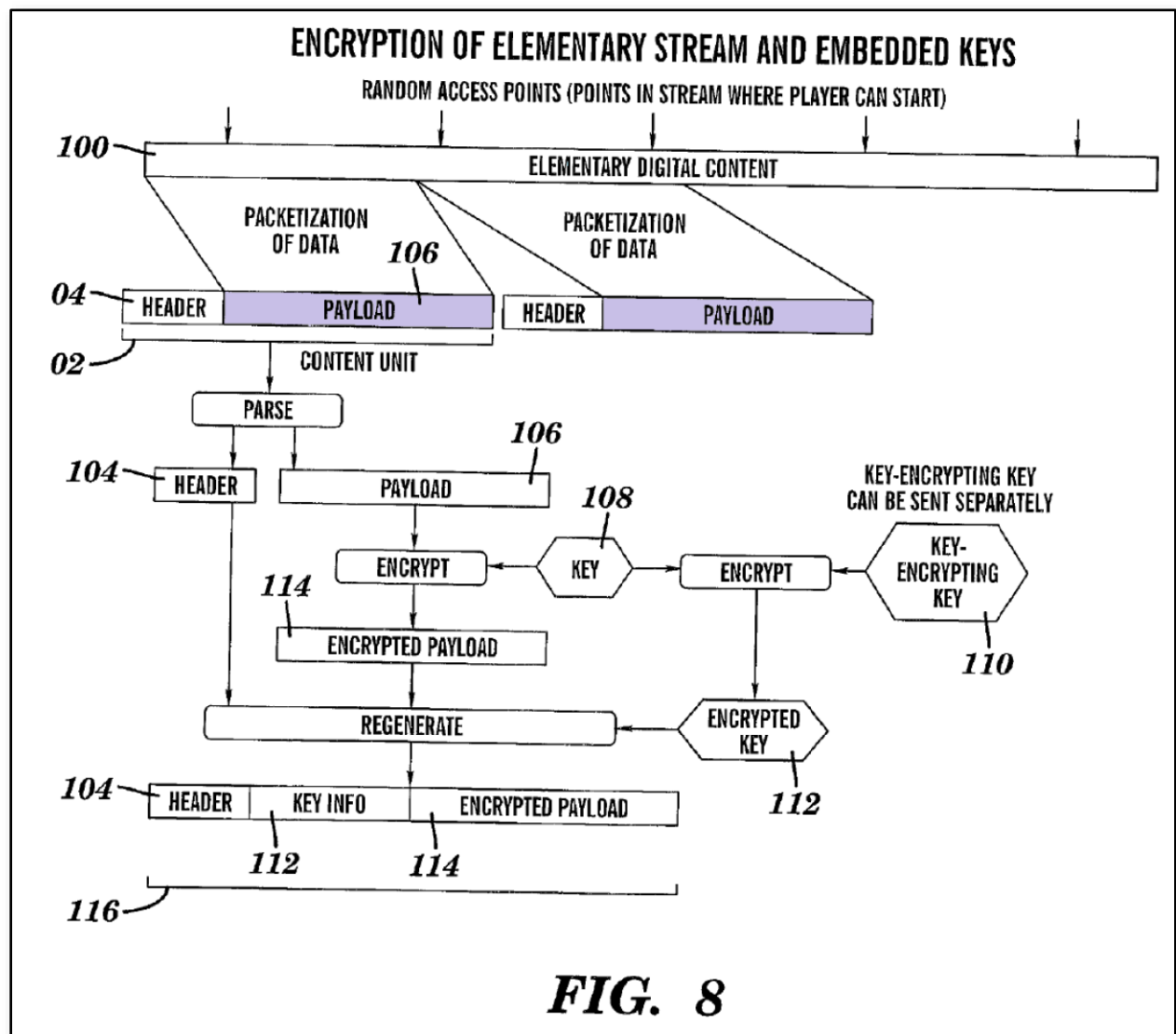
As discussed below, Foster “*generat[es] a plurality of data chunks based on a data set, wherein each data chunk of the plurality of data chunks comprises less than an entirety of data of the data set,*” as claimed.

As illustrated in Foster’s Figure 8 (annotated below), Foster generates a plurality of “**payloads**” (“***data chunks***”) by “*parcel[ing]*” the elementary digital content. *See also supra* §VII.B.1 (1[PRE]); Foster, [0064]; Bhattacharjee, ¶¶183-84.

Foster’s digital content is a “*data set*” under both parties’ constructions because POSAs understood that the digital content (*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). Foster, [0004], [0008], FIG. 8, [0037] (“Content—any data such as digital

image, sound or binary data deliverable from a source to a recipient”), [0050]; Bhattacharjee, ¶185.

As Foster’s Figure 8 (color-annotated below) illustrates, each **payload** (“**data chunk**”) “comprises less than an entirety of the data set,” as claimed, because Foster discloses that the entire stream (“**data set**”) cannot be recovered from any one single payload. Foster, [0071]; Bhattacharjee, ¶186.



5. [1C2]

As discussed below, Foster meets “*the data set can be reconstructed using at least a minimum number of the plurality of chunks,*” as claimed, under SFI’s proposed constructions and renders it obvious under Google’s proposed constructions.

Foster 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 (encrypted “*data chunks*”) are required in order to access (“*reconstruct*”) the entire stream (“*data set*”). Therefore, the elementary content stream (the claimed “*data set,*” §VII.B.4, [1C1]) “*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, 9; *see also* EX1009, 4-5; Bhattacharjee, ¶¶188-89.

Under Google’s construction, POSAs had reason to distribute Foster’s “*data set*” into encrypted payloads (encrypted “*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, ¶¶190-91 (citing EX1050, 1:19-22, 2:31-38, 11:11-26, 15:39-16:20; EX1052, [0001]).

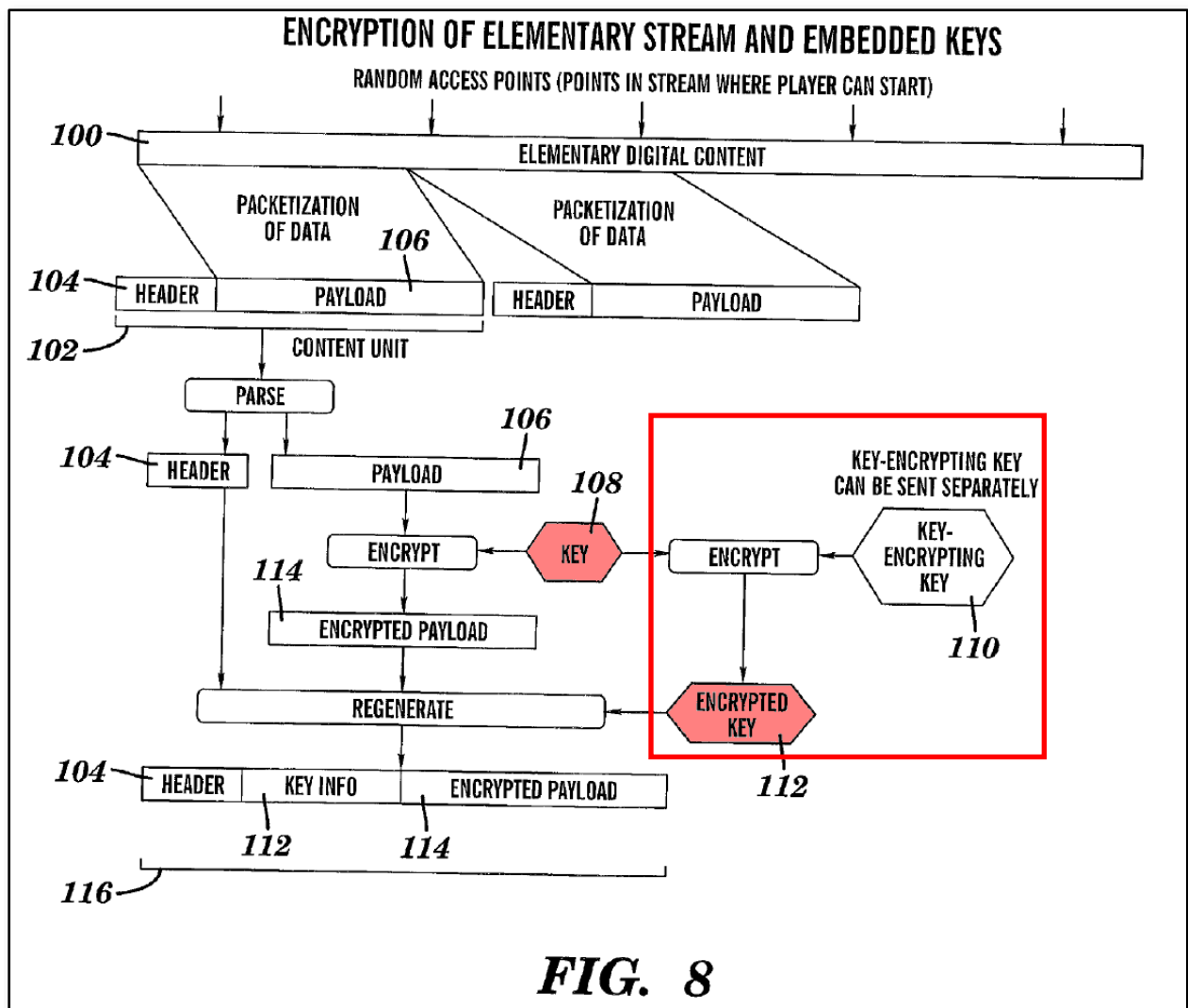
This implementation of Foster’s system would have been a conventional way to distribute and store data by the early 2000s and it had known benefits – like redundancy and protection against loss of / inability to retrieve individual data chunks. Bhattacharjee, ¶¶190-91 (citing EX1050, 1:19-22, 2:31-38, 11:11-26, 15:39-16:20). That distribution technique would have also been applying a known data-distribution technique to a known device (Foster’s encryption device) ready for the improvement of being able to recover data from fewer than all chunks. Bhattacharjee, ¶192.

POSAs would have had a reasonable expectation of success as the above data-distribution techniques were well-known, conventional, and based on precise mathematics. Bhattacharjee, ¶193 (citing as corroboration EX1050, 1:19-22, 2:31-38, 11:11-26, 16:6-20; EX1052, [0001]).

6. [1D]

As discussed below, Foster “*encrypt[s] each respective data chunk of the plurality of data chunks with a respective second key, wherein each of the respective second keys are distinct from each other,*” as claimed.

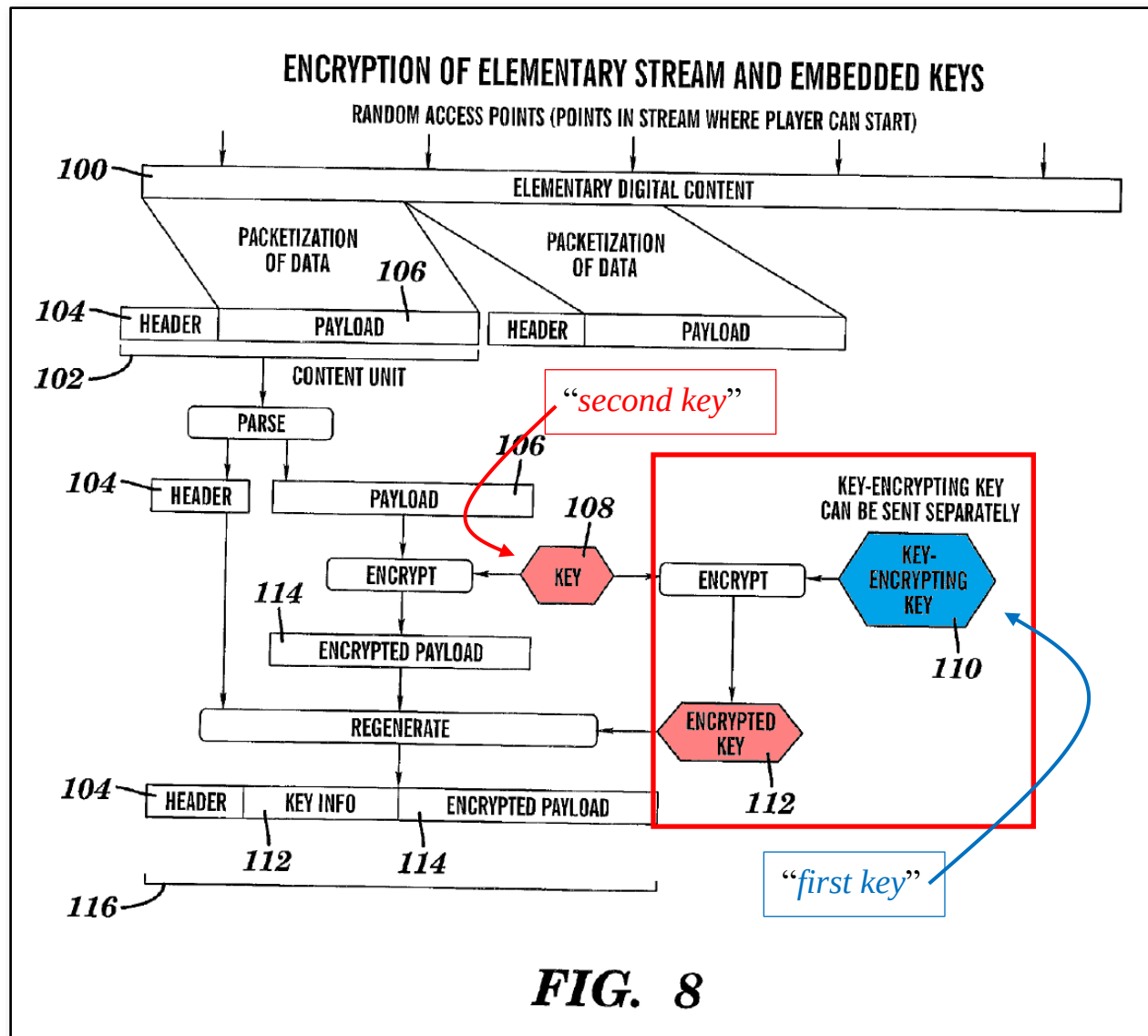
In Foster, “each content unit” (FIG. 8’s payload, which is a “*data chunk*,” as claimed, *supra* §VII.B.4) “could be *encrypted* with a *different* title key.” Foster, [0064]-[0065], [0071]. Each “different title key” is a “*respective second key*” and each one is “*distinct from each other*,” as claimed, because Foster discloses that they are “different” (Foster, [0064]-[0065] and that a “pirated [title] key is only good for decrypting the content packet to which is was attached” (Foster, [0071]); Bhattacharjee, ¶¶194-95.



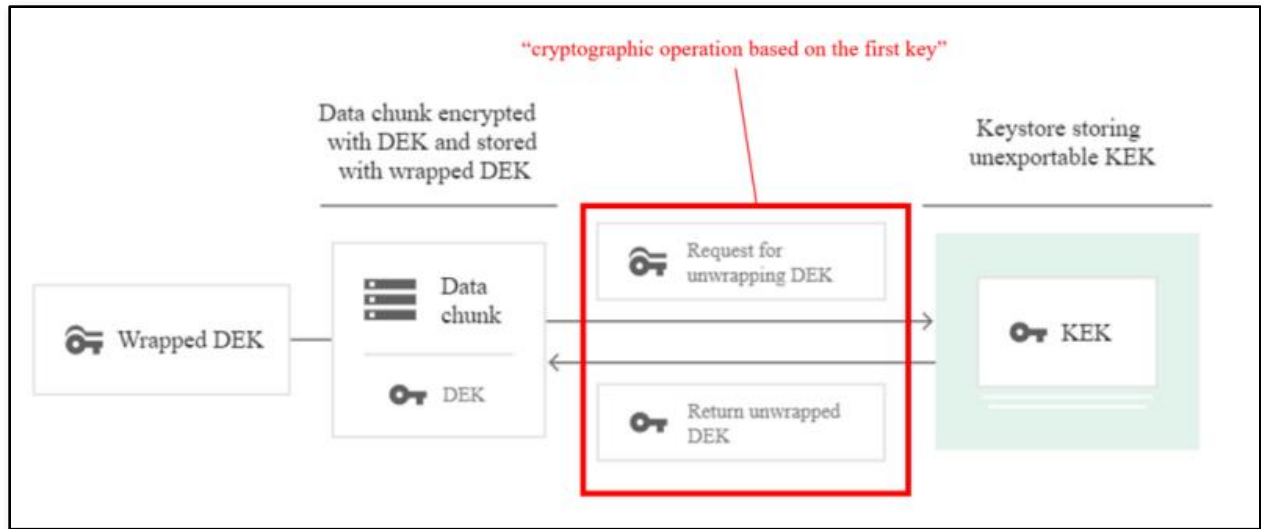
7. [1E]

As discussed below, Foster “*perform[s] a cryptographic operation based on the first 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 “*first key*,” see §VII.B.3, [1B]), which is “*a cryptographic operation based on the*” KEK. Foster, [0064]-[0065]; Bhattacharjee, ¶¶196-97.



As shown below, under the scope that SFI attributes to [1E] in its district-court infringement contentions, encrypting [1D]’s title keys with a KEK meets [1E]. See EX1009, 8-9 (arguing “DEKs [data encryption keys] are encrypted with (wrapped by) a key encryption key (KEK)”); §VII.B.6. Thus, Foster meets [1E] under SFI’s assertion of the claim scope. Bhattacharjee, ¶198.



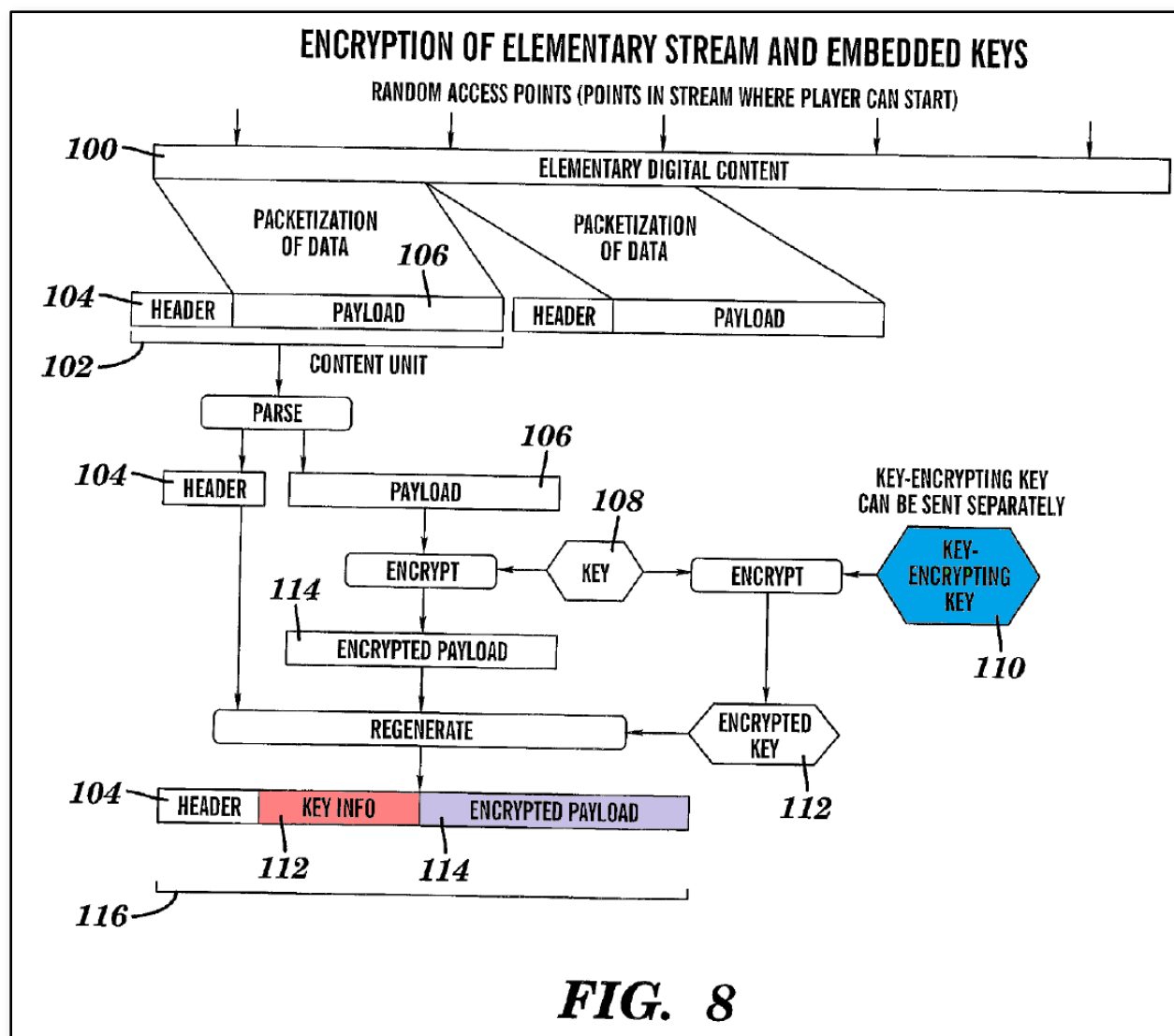
EX1009 (SFI’s Infringement Contentions), 9

Alternatively, [1E] is met because POSAs also had motivation for (and expected success in) reordering data within each encrypted “*data chunk*” based on the “*first 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.E. Bhattacharjee, ¶199 (citing EX1028, EX1031, EX1032, EX1040, EX1041, EX1028, EX1045, EX1054).

8. [1F]

As discussed below, Foster discloses and renders obvious “*storing, in a memory coupled to the processor, at least one data chunk of the plurality of data chunks with data indicative of at least one of the distinct encryption keys on at least one storage device,*” 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 (an encrypted “*data chunk*”); (2) “encrypted **title key** 112” (a “*distinct encryption key*”), which is a **title key** “after it has been encrypted with the **KEK**” (the “*first 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; Bhattacharjee, ¶¶200-01. 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) can store the content units **because** each includes a “*storage device*” (*e.g.*, a hard drive). Foster, [0058] (“receiver may **cache**” (store) “KMBs...in **memory**” (a storage device)); Bhattacharjee, ¶201.



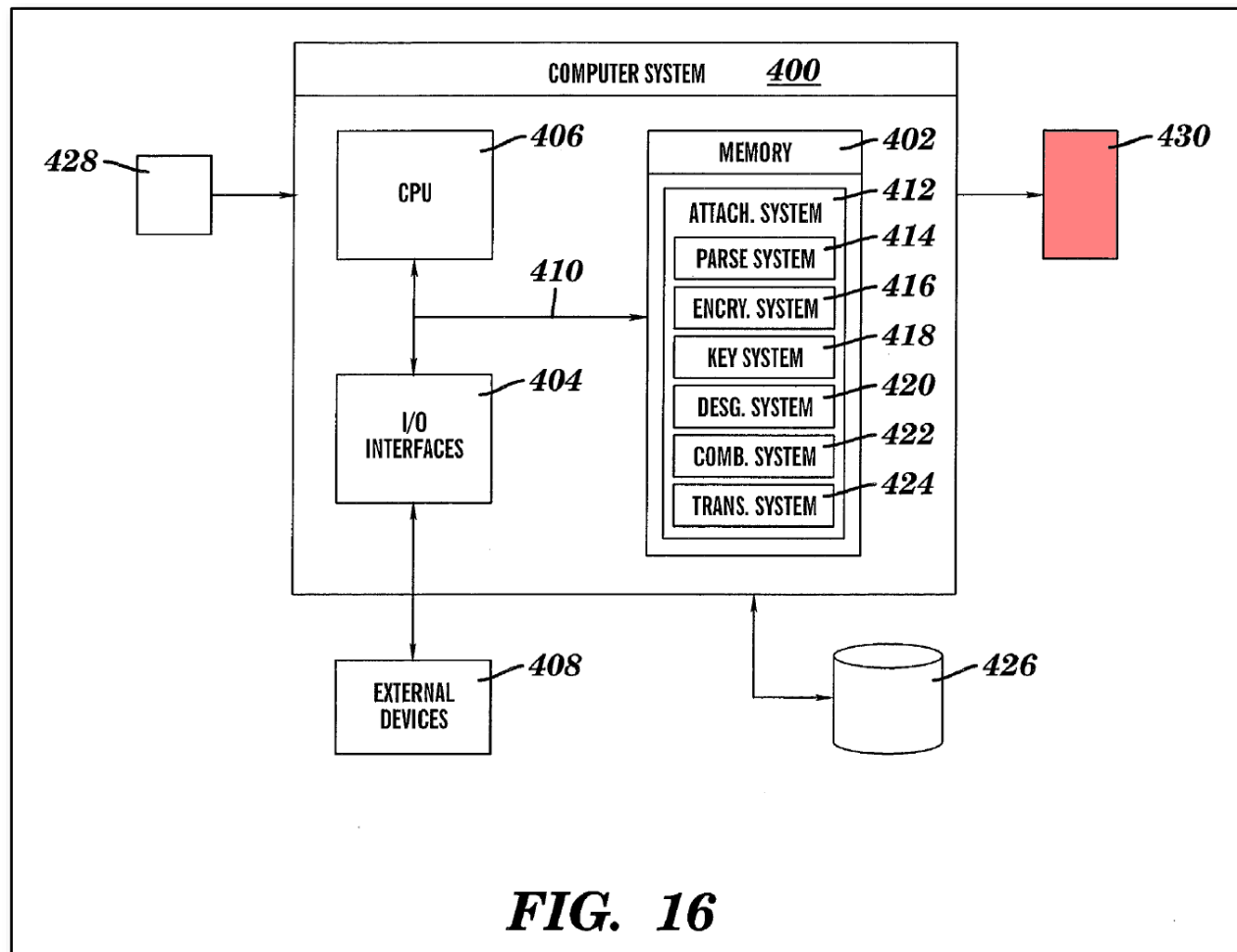
Under SFI’s claim constructions and the scope SFI attributes to [1F] in related district- court litigation, [1F]’s “data indicative of at least of the distinct encryption keys” is met by one of [1D]’s keys itself. EX1009, 10-11 (alleging limitation is met by “stor[ing] each DEKs [data encrypting keys alleged to meet [1D]] with the data chunks”); Bhattacharjee, ¶202.

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 key length for each

title key, which is “*data indicative of*” a title key under Google’s construction. EX1064, 5-6; Bhattacharjee, ¶203 (citing EX1039).

As discussed above (§VI.B.8), POSAs understood the ’609 patent uses “coupled to” broadly to include a connection using any communication interface that allows two components to exchange information. EX1001, 66:49-55 (describing a component “**coupled to [a] storage area [] using any suitable communication interface []** (e.g., USB, serial, parallel, Bluetooth, IR, IEEE 1394, Ethernet, **or any other suitable communications interface**”); Bhattacharjee, ¶204.

Each content unit 116 discussed above is stored on receiver at “receipt 430”; that receiver is a “*storage device*” (e.g., consumer set-top box) with “*memory*” (e.g., a hard drive). Foster, [0040], [0052], [0058]. As illustrated in Figure 16 (color-annotated below) recipient 430 (and its memory) is in communication with CPU 406 (the “*processor*”) via “transmission system 424” and “bus 410.” Foster, [0078]-[0079]. Thus, the memory in recipient 430’s storage device is “*coupled to*” Foster’s CPU (“*processor*”). They are also coupled to each other under the claim scope that SFI asserts for [1F] in related district-court litigation. EX1009, 10-11 (alleging limitation is met by storage in “storage systems” remote from the alleged processor performing the encryption); Bhattacharjee, ¶205.



Alternatively, POSAs also had reason to store Foster’s encrypted content in the memory of storage devices coupled to the processor (*e.g.*, through a bus, direct wired/wireless connection, and/or a network-based connection). As discussed above (§VII.A), Foster’s system 400 is (or can be part of) a content service provider. As illustrated in Figure 16 (above), Foster’s system 400 includes memory 402, which comprises “any known type of data storage,” including “one or more types of data storage” or “distributed [storage] across a plurality of physical systems.” Foster,

[0075]. Thus, Foster’s memory 402, which is coupled to CPU 406 via bus 410, includes storage devices. Bhattacharjee, ¶206.

POSAs had reason to use one of the conventional storage devices coupled to Foster’s processor to store data sets encrypted by Foster’s processor. Doing so would have beneficially allowed Foster’s system to store copies of encrypted content for (1) backup, (2) video-on-demand distribution, and/or (3) distribution to later-identified recipients (*e.g.*, new subscribers), without having to repeat the encryption processing every time digital content (*e.g.*, a movie) needed to be encrypted and delivered to recipients. Bhattacharjee, ¶¶207-11 (citing as corroboration EX1045, FIGS. 5-6, [0109]-[0111]; EX1050, 1:19-22, 2:31-38, 11:11-26, 16:6-20; EX1059, 2:9-4:10, 5:4-23; EX1060, [0007]-[0008]).

Storing encrypted content in memory coupled to Foster’s processor on at least one storage device would have also been applying a known technique to solve the known problem of how to store encrypted content for delivery to multiple recipients at varying times, and thus was obvious. *Intel*, 61 F.4th at 1380 (“[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, ¶212 (citing EX1045).

Thus, Foster discloses and renders obvious storing each “processed content unit 116” – a distinct title key (“*distinct encryption key*”) and respective encrypted

payload (encrypted “*data chunk*”) – “*in a memory coupled to the processor... on at least one storage device,*” as claimed. Bhattacharjee, ¶213.

As discussed above (§§VII.B.4-VII.B.8), Foster performs limitations [1B]-[1F] 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, ¶214.

C. Claim 2

Foster meets claim 2 for the reasons it meets [1F] as discussed above (§VII.B.8). Specifically, Foster stores with each “*data chunk*” (Foster’s payload) the title key (one of [1D]’s “*distinct encryption keys*”) used to encrypt that data chunk. Foster, [0065]-[0066], FIGS. 8-9. As discussed above (§VII.B.8), a title key is “*data indicative of at least of the distinct encryption keys*” under SFI’s constructions and assertion of the claim scope in related district-court litigation. Bhattacharjee, ¶¶215-16; EX1065, 3-4; EX1009, 11-13.

Alternatively, as also discussed above (§VII.B.8), POSAs had reason to store with each data chunk metadata about a title key (*e.g.*, its length and/or associated encryption mode), which is “*data indicative of*” a title key (“*distinct encryption key*”) under Google’s construction. EX064, 5-6; Foster, [0072] (embodiments with variable packet length); Bhattacharjee, ¶27 (citing EX1039, EX1056).

D. Claim 3

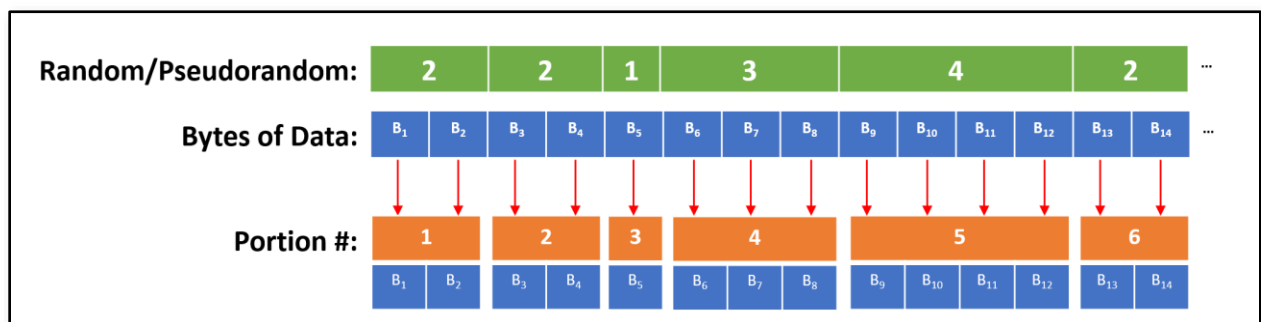
Foster meets claim 3 for the reasons it meets [1F] (§VII.B.8) and claim 2 (§VII.C). Specifically, Foster stores with each “*data chunk*” (Foster’s payload) the title key (one of [1D]’s “*distinct encryption keys*”) used to encrypt *that* data chunk. Foster, [0065]-[0066], FIGS. 8-9. As discussed above (§VII.B.8), a title key is “*data indicative of at least of the distinct encryption keys*” under SFI’s constructions and assertion of the claim scope in related district-court litigation. Bhattacharjee, ¶218; EX1065, 3-4; EX1009, 11-13.

Alternatively, as also discussed above (§VII.B.8), POSAs had reason to store with each data chunk metadata about the chunk’s title key (*e.g.*, its length and/or associated encryption mode), which is “*data indicative of*” that chunk’s title key under Google’s construction. EX064, 5-6; Bhattacharjee, ¶219 (citing EX1039; EX1056; EX1064; Foster, [0072]).

E. Claim 4

Below illustrates three well-known, conventional techniques for generating Foster’s payloads (“*data chunks*”) “*using a random technique or pseudorandom technique*” that would have been obvious in Foster to provide additional security (*e.g.*, by making it harder for attackers to analyze/predict the structure of encrypted data chunks), in view of POSAs’ background knowledge evidenced by multiple references cited below.

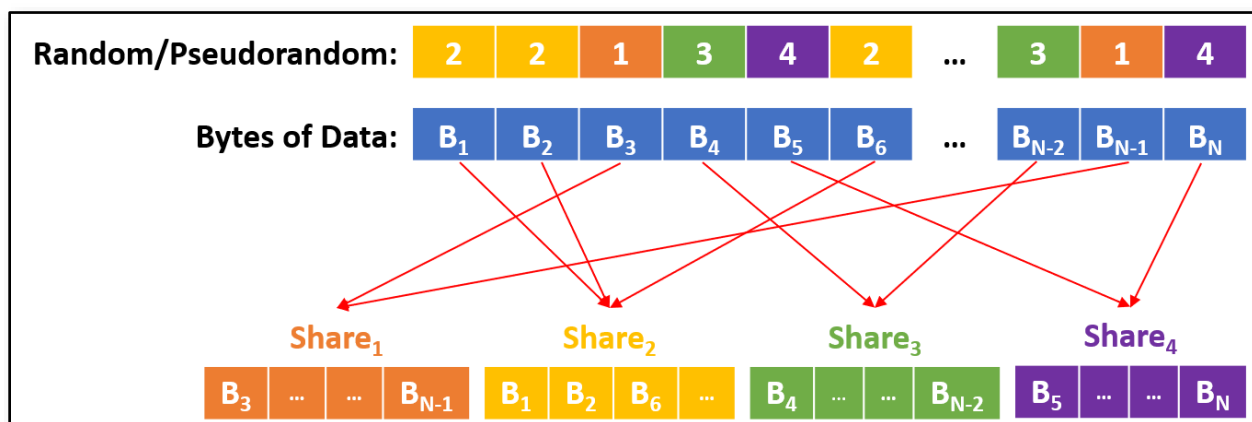
First, POSAs had reason to divide the data set into random- or pseudorandom-sized payload portions (as illustrated below) as a well-known conventional technique to provide an additional security (*e.g.*, by making it harder for attackers to analyze/predict the structure of encrypted data chunks). Bhattacharjee, ¶¶220-22 (citing EX1044, [0016]-[0018], [0077]; EX1045, FIGS. 9-10, [0103]-[0108]; EX1046, [0003], [0021]; EX1028, [0015]-[0017]); Foster, [0072] (disclosing embodiment where packet length is not fixed).



Second, as discussed above (§VII.B.7), POSAs also had reason to randomly/pseudorandomly generate the payloads (“*data chunks*”) by rearranging

data within each payload, also well-known for additional security benefits. Bhattacharjee, ¶223 (citing EX1028, EX1044-EX1046).

Third, it was well-known to POSAs that generating payloads (“*data chunks*”) by distributing bits/bytes of the data set into payloads according to a random/pseudorandom number (which is using a random/pseudorandom encryption key, as illustrated below) provided additional security when generating the payloads (“*data chunks*”). Bhattacharjee, ¶¶224-25 (citing EX1047, [0056]-[0058]; EX1048, [0008]; EX1045, FIGS. 9-10, [0103]-[0108]; EX1049, 10:8-15; EX1061, [0059]).



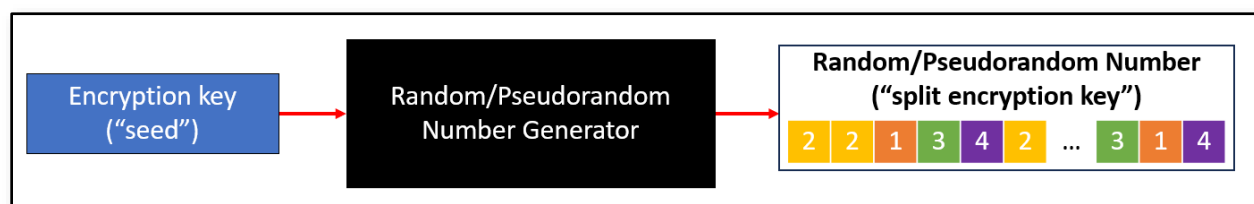
F. Claim 5

Foster meets claim 5 because the random/pseudorandom techniques discussed for claim 4 result in a substantially random distribution of a subset of the data set in each payload (“*chunk*”), as discussed *supra* §VII.E (Claim 4). Bhattacharjee, ¶226 (citing EX1028, EX1044-EX1049).

G. Claim 6⁸

As discussed above (§VII.E, Claim 4), POSAs had reason to generate Foster’s content units by distributing the data set into payloads (“*data chunks*”) data chunks according to a random/pseudorandom number. For example, as discussed, the random/pseudorandom number would determine the payload (“*data chunk*”) sizes and/or what payload receives a particular bit (byte or other portion) of the data set. See §VII.B.7 (citing corroboration). When used to distribute in one of those manners, the random/pseudorandom number meets a “*split encryption key*” because its sequence is used in the algorithm for distributing Foster’s data into data chunks, as discussed above (§VI.G). Bhattacharjee, ¶227.

As also discussed above (§VI.G), it was conventional to generate random/pseudo-random numbers using an encryption key – as a so-called “seed” for a number generator. Bhattacharjee, ¶228 (citing as corroboration EX1040, [0030]; EX1041, [0024], [0026]-[0027]).



⁸ Claims 6-10 as addressed herein account for SFI’s granted certificate of correction, which replaced “*external*” with “*first*.” See EX1002, 12-16; §IV.C.

Because Foster’s KEK (the “*first key*”) is an encryption key used to secure the data set, POSAs had reason to also use Foster’s KEK as the seed to generate the above-discussed “*split encryption key*.” Bhattacharjee, ¶229.

Alternatively, POSAs had reason to use another conventional encryption key as the “*split encryption-key*” (or as the seed to generate that key), to achieve the same benefits discussed above. Bhattacharjee, ¶229 (citing EX1051, [0037]).

When using an additional key as or to generate the split encryption key, POSAs had reason for the key(s) to be stored and retrieved from Foster’s database 426 (a storage system) because database 426 was already designed to store encryption keys (see §VII.B.3, [1B]). Thus, a conventional encryption key would have been an alternative “*first key*” to use in Foster’s encryption method and POSAs had reason to beneficially use that key as a split encryption key or as a seed to generate a split encryption key, for the reasons discussed above in this section. Bhattacharjee, ¶230.

Accordingly, Foster renders claim 6 obvious. Bhattacharjee, ¶231.

H. Claim 7

As discussed above (§VII.B.7, [1E]), Foster encrypts the title key for each payload (the claimed “*data chunk*”) with a KEK (the claimed “*first key*”). SFI has asserted that encrypting a data chunk’s encryption key with a KEK is “further secur[ing]” the data chunk. EX1009, 8-9. Thus, by encrypting each payload’s title

key with a KEK, Foster is also “*encrypting each of the plurality of chunks with the first key*,” as claimed, under SFI’s assertion of claim scope. Bhattacharjee, ¶232.

If claim 7 more narrowly requires encrypting data chunks directly with the “*first key*,” doing so by encrypting Foster’s data set using Foster’s KEK or an alternative (conventional) encryption key retrieved from a storage system (*e.g.*, Foster’s database 426) would have been an obvious implementation of Foster. POSAs would have had motivation and expected success in doing so to provide additional security to Foster’s encrypted payloads (encrypted “*data chunks*”) beyond encrypting title keys with Foster’s KEK. Bhattacharjee, ¶233.

As also discussed above (§VI.B.7.b.iii), POSAs had reason to use a cryptographic operation based on an encryption key to randomize the data within each portion of the data set (“*data chunk*”). To generate the random/pseudorandom number used for that randomization, it was conventional to use an encryption key itself or as a “seed” to a number generator. §VI.G (citing corroboration). For that key, POSAs had reason to use Foster’s KEK or another conventional key received from a storage device, thereby encrypting each chunk with that “*first key*.” Bhattacharjee, ¶¶234-36.

I. Claim 8

Foster meets claim 8 for the same reason it meets limitation [1C2] (§VII.B.5) and Claim 7 (§VII.H). Bhattacharjee, ¶237.

J. Claims 9-10

As discussed above (§VII.B.7, [1E]), Foster encrypts title keys with a key-encryption-key (a “*first key*”). Foster discloses that the KEK is used for encryption and decryption. Foster, [0054]-[0055]. Thus, Foster’s KEK is a so-called symmetric encryption key. Bhattacharjee, ¶238; EX1001, 32:57-61 (noting that in “symmetric key cryptographic systems” a “single common key [] is used to encrypt and decrypt a message”).

POSAs had reason to implement Foster’s (symmetric) KEK as an AES- or DES-compliant key because those were well-known, conventional encryption standards for symmetric encryption keys that were published years before the earliest-potential priority date, and so using either would have had the benefit of using well-tested and well-understood encryption techniques, as discussed above §VII.J. Bhattacharjee, ¶239 (citing as corroboration EX1057, [0006]; EX1058, 1:12-48, 4:52-54).

For similar reasons, implementing Foster’s KEK as an AES or DES key would have been using a known encryption key technique (based on well-known encryption standards) to improve similar devices (for encrypting data) in the same way (by using an established technique for generating encryption keys). *KSR*, 550 U.S. at 417; Bhattacharjee, ¶240.

Therefore, Foster renders obvious claims 9 and 10.

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 (SFI’s August 14, 2023 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: The Petition Advances Art and Arguments Not Previously Considered

Because the Office never previously considered the challenged claims' patentability in view of Orsini or Foster, Step 1 of *Advanced Bionics* is not satisfied and there is no need to proceed to Step 2.

2. Step 2: The Office Erred Materially

Orsini issued as U.S. Patent No. 7,391,865, which SFI cited in an IDS during prosecution of SFI's '854 patent (which is a parent of the '609 patent and is being challenged in co-pending IPR2024-00215). Although the '865 patent was identified as "considered" during prosecution of the '854 patent, the examiner never discussed or rejected any claims of the '854 patent over the Orsini '865 patent. If the Board finds Step 1 satisfied merely because Orsini's issued patent was cited in an IDS during prosecution of the '854 patent, 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 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).

Petitioner is not asking the Office to reconsider any arguments regarding claims 1-10 in view of Orsini; they are asking the Office to consider whether Orsini anticipates and/or renders obvious those claims *for the first time*.

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 renders claims 1-10 unpatentable. This is “something persuasive” that the Office overlooked when it allowed the claims despite being aware of Orsini’s issued patent.

IX. CONCLUSION

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

Dated: 2023-11-22

Respectfully submitted,
Google LLC

/Gregory Nieberg/
Gregory S. Nieberg, Reg. No. 57,063
WOLF, GREENFIELD & SACKS, P.C.

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 13,947 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.

X. APPENDIX: CLAIM LISTING⁹

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 securing data, the method comprising:
[1A] executing code by a processor to perform:
[1B] receiving a first key from a storage system;
[1C1] generating a plurality of data chunks based on a data set, wherein each data chunk of the plurality of data chunks comprises less than an entirety of data of the data set, and
[1C2] wherein the data set can be reconstructed using at least a minimum number of the plurality of chunks;
[1D] encrypting each respective data chunk of the plurality of data chunks with a respective second key, wherein each of the respective second keys are distinct from each other;
[1E] performing a cryptographic operation based on the first key to further secure the plurality of data chunks; and
[1F] storing, in a memory coupled to the processor, at least one data chunk of the plurality of data chunks with data indicative of at least one of the distinct encryption keys on at least one storage device.
Claim 2
2. The method of claim 1, further executing the code by the processor to perform: storing with each data chunk of the plurality of data chunks data indicative of at least one of the distinct encryption keys.
Claim 3
3. The method of claim 1, further executing the code by the processor to perform: storing with each data chunk of the plurality of data chunks data indicative of a distinct encryption key used to encrypt that data chunk.
Claim 4
4. The method of claim 1, wherein generating the plurality of data chunks comprises using a random technique or pseudorandom technique.

⁹ This Claim Listing includes corrections to claims 6-10 identified in SFI's granted certificate of correction. EX1002, 12-16; *see also* §IV.C.

Claim 5
5. The method of claim 1, wherein generating the plurality of chunk [sic] comprises a substantially random distribution of a subset of the data set.
Claim 6
6. The method of claim 1, wherein generating the plurality of data chunks comprises: generating a split encryption key based on the first key; and distributing the data set based on the split encryption key.
Claim 7
7. The method of claim 1, further executing the code by the processor to perform: encrypting each of the chunks of the plurality of chunks with the first key.
Claim 8
8. The method of claim 7, wherein the data set is recoverable using at least a minimum number of the plurality of data chunks and the first key.
Claim 9
9. The method of claim 1, wherein the first key is an AES key.
Claim 10
10. The method of claim 1, wherein the first key is an DES key.