

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

REPLY DECLARATION OF SAMRAT BHATTACHARJEE, PH.D.

Google Exhibit 1111 Google v. SFI

TABLE OF CONTENTS

I.	Materials Reviewed And Considered	1
II.	My Reply to SFI Regarding Ground 2: IT IS STILL MY OPINION THAT ALL CHALLENGED Claims WOULD HAVE BEEN ANTICIPATED BY AND/OR RENDERED OBVIOUS TO POSAs over Foster.....	3
	A. Response to Argument that Foster Is Not in the context of securing data (using at least encryption) for storage.	5
	B. Response to Argument that Foster’s Encrypted Payloads Are Not <i>Data Chunks</i>	25
	C. Response to Argument that Foster Does not Disclose or Render Obvious the Processor Executing Code To Store at least one Data Chunk on a Memory Coupled to the Processor ([1A], [1F])	36
	1. It is still my opinion that Foster Discloses that its Processor is Executing Code to Store at least one Data Chunk on a Memory Coupled to Foster’s Processor	36
	2. It is still my opinion that Foster Renders Obvious Executing Code to Store at least one Data Chunk on a Memory Coupled to Foster’s Processor	49
	D. Response to Argument that that Foster Does not Disclose or Render Obvious “Coupl[ing] to the Processor” Storage-Device Memory ([1F]).....	53
	1. It is still my opinion that Foster Discloses “Coupl[ing] to the Processor” Storage-Device Memory	53
	2. It is still my opinion that Foster Renders Obvious “Coupl[ing] to the Processor” Storage-Device Memory	58
III.	My Reply to SFI Regarding Ground 1: IT IS STILL MY OPINION THAT ALL CHALLENGED Claims WOULD HAVE BEEN OBVIOUS over Orsini	85
	A. Response to Argument that it Would Not Have Been Obvious to POSAs to Encrypt Orsini’s Share Keys Using a Known Key- Encrypting Key (“KEK”).....	87
	1. My KEK-based Obviousness Opinion was Independent of, and based on independent corroborating evidence from, my	

opinion that Orsini’s public-key encryption disclosures provided an anticipating embodiment.	106
2. My KEK-based Obviousness Opinions Did Not Rely on Common Sense to Supply a Missing Limitation in the Prior Art.	107
a. POSAs would not have unagreed that limitation [1E] was missing in the prior art.	108
b. My opinions on the obviousness of encrypting Orsini’s share keys with a KEK was not based on common sense.	110
3. My reply to SFI’s Challenges to my opinions regarding POSAs’ Motivation-to-Combine Orsini with a conventional KEK	112
a. It is still my opinion that POSAs Would Have Wanted to Make Orsini’s System More Secure by using a KEK.	112
b. In my opinion, any alleged Drawbacks to the modifying Orsini to use a KEK that SFI Identifies would have been Easily Overcome	119
c. My opinion KEK-based Obviousness Opinion was not based on hindsight, but was instead grounded in evidence that I cited.....	124
B. Response to Argument that POSAs would not have a reasonable expectation of success in modifying Orsini to meet limitation [1E]	126

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

1. I have been retained by Wolf, Greenfield & Sacks, P.C., counsel for Petitioner Google LLC, to assess claims 1-10 (the “challenged claims”) of U.S. Patent No. 11,068,609 (“the ’609 patent”) (EX1001). I am being compensated for my time at my standard rate of \$700.00 per hour, plus actual expenses. My compensation is not dependent in any way upon the outcome of the *inter partes* review of the ’609 patent.

I. MATERIALS REVIEWED AND CONSIDERED

2. My findings, as explained below, are based on my years of education, research, experience, and background in field of securing data from unauthorized access or use, as well as my investigation and study of relevant materials for this declaration. *See also* EX1070 (Supplemental Declaration of Samrat Bhattacharjee, Ph.D.). When developing the opinions set forth in this declaration, I assumed the perspective of a person having ordinary skill in the art (“POSA”), as set forth in Section IV of my initial declaration submitted in this proceeding (EX1003, “First Declaration”).

3. In forming my opinions set forth in this declaration, in addition to the references I discussed and reviewed for my First Declaration, I have reviewed SFI’s POR (Paper 22), Dr. Rubin’s declaration (EX2033), the exhibits cited by SFI and Dr. Rubin, and the following exhibits and papers as well as any papers or

exhibits explicitly cited herein that I cite in response to SFI's POR and/or Dr.

Rubin's declaration:

Exhibit	Description
1072	U.S. Patent Application Publication No. 2004/0019900
1073	U.S. Patent Application Publication No. 2002/0083438
1074	U.S. Patent No. 7,076,661
1075	U.S. Patent No. 6,771,657
1076	U.S. Patent No. 6,389,473
1077	U.S. Patent No. 6,662,329
1078	International Application Publication No. WO 2003/026220
1079	Trappe, Key Management and Distribution for Secure Multimedia Multicast (IEEE Dec. 2003)
1080	D. McGovern, "Pushing the internet direct to the user - security issues," IEE Seminar on IP Over Satellite - The next Generation: MPLS, VPN and DRM Delivered Services, 2003., 2003, pp. 122-138, doi: 10.1049/ic:20030063.
1081	U.S. Patent No. 7,277,941
1082	Mike Knee, MPEG Video.
1083	U.S. Patent No. 7,266,613
1084	U.S. Patent Application Publication No. 2004/0202320
1085	U.S. Patent No. 7,823,009
1086	U.S. Patent No. 8,098,819
1087	U.S. Patent Application Publication No. 2004/0236958
1088	International Application Publication No. WO 2001/015162
1089	U.S. Patent Application Publication No. 2004/0107356
1090	U.S. Patent No. 7,739,501
1091	U.S. Patent Application Publication No. 2004/0162871
1092	U.S. Patent Application Publication No. 2002/0129383
1093	U.S. Patent Application Publication No. 2004/0005061
1094	U.S. Patent Application Publication No. 2005/0160292
1095	U.S. Patent No. 6,959,086
1096	RFC 3394: Advanced Encryption Standard (AES) key wrap algorithm
1097	U.S. Patent No. 7,257,227
1098	U.S. Patent No. 6,993,508
1099	Deposition of Aviel D. Rubin, Ph.D. (Dec. 12, 2024)
1100	U.S. Patent Application Publication No. 2003/0140257

1101	U.S. Patent Application Publication No. 2004/0199771
1102	U.S. Patent Application Publication No. 2005/0177740
1103	U.S. Patent Application Publication No. 2004/0199667
1104	U.S. Patent Application Publication No. 2005/0050218
1105	U.S. Patent Application Publication No. 2003/0187799
1106	National Institute of Standards and Technology, NIST Special Publication 800-57, Recommendations for Key Management – Part 1: General (Aug. 2005)
1107	U.S. Patent Application Publication No. 2004/0236958
1108	U.S. Patent No. 6,169,843
1109	MPEG-2 - a tutorial introduction to the systems layer (IEEE 1995)
1110	U.S. Patent Application Publication No. 2003/0145329
1111	Reply Declaration of Samrat Bhattacharjee, PH.D.
1112	U.S. Patent Application Publication No. 2004/0010699
1113	European Patent Application Publication No. EP1,564,686
1114	U.S. Patent Application Publication No. 2004/0010699
1115	U.S. Patent Application Publication No. 2004/0187014
1116	U.S. Patent Application Publication No. 2004/0216164 (“Hayhurst”)
1117	U.S. Patent Application Publication No. 2006/0143350

II. MY REPLY TO SFI REGARDING GROUND 2: IT IS STILL MY OPINION THAT ALL CHALLENGED CLAIMS WOULD HAVE BEEN ANTICIPATED BY AND/OR RENDERED OBVIOUS TO POSAS OVER FOSTER¹

4. As I discussed in my First Declaration (¶162), Ground 2 is based on the Foster reference (EX1006). In particular, it was and still is my opinion Foster’s encryption methods anticipate and/or render obvious all challenged claims of the ’609 patent. First Declaration, ¶¶162-240. As I discussed in my First Declaration

¹ I have added all emphases added throughout this declaration unless I have noted otherwise.

(¶¶164-173), Foster divides a *dataset* (e.g., a movie file) into segments (which Foster calls “payloads”), encrypts the segments with respective keys, encrypts those keys with a key-encryption key (KEK), and stores all encrypted segments plus corresponding encryption keys on a storage device. *E.g.*, Foster, [0064]-[0072], FIG. 8. As I discuss below (§II.C.2, §II.D.2), for my opinions on obviousness over Foster, my opinion regarding POSAs’ motivation to modify Foster and their reasonable expectation of success in doing so is reinforced by the corroborating evidence I cited in my First Declaration and the additional corroborating evidence I cite in this declaration in response to SFI’s arguments and Dr. Rubin’s opinions.

5. As I discuss further below (§§II.A-II.C), in reviewing SFI’s POR and the Rubin Declaration, SFI appears to have three primary arguments challenging how Foster anticipates and renders obvious the challenged claims: (1) SFI argues that Foster does not disclose claim 1’s *data chunks* (POR, 34-42); (2) SFI’s argues that Foster does not disclose or render obvious executing code to store at least one data chunk on a memory coupled to a processor (POR, 42-48); and (3) SFI argues that Foster does not disclose or render obvious coupling a processor to the memory of a storage device (POR, 48-56). As I discuss in the sections below (§§II.A-II.B), in my opinion POSAs would disagree with each of these arguments.

A. Response to Argument that Foster Is Not in the context of securing data (using at least encryption) for storage.

6. SFI argues the '609 and Foster “arise in entirely different context, are directed towards addressing fundamentally different problems, and result in very different inventions.” POR, 32. SFI’s arguments only consider Foster in the context of streaming live television broadcasts, and argues that the '609 patent is about storing encrypted content. POR, 32-34. I disagree and in my opinion POSAs would disagree as well.

7. As an initial matter, as I discussed in my First Declaration (and further below, §II.B, §II.C.1, §II.D.1), it is my opinion that Foster anticipates independent claim 1 – the only claim SFI addresses in its POR. Thus, the alleged differences in contexts, problems, and inventions that SFI discusses in its POR are irrelevant because Foster teaches embodiments that meet every limitation of those claims.

8. I also disagree that Foster is an entirely different context or addresses entirely different problems than the '609 patent. SFI argues in the related IPR matter for the '854 patent – which shares a written description with the '609 patent – that SFI’s patent family is in the field of securing data and addresses at least the problem of providing user independent security. IPR2024-00215, Paper 21, 33.

38. As I discussed in my First Declaration and again below (§II.B), Foster splits a data set into segments, encrypts each of the segments with a respective encryption key, and encrypts each of those encryption keys with a key-encryption key

(“KEK”). Thus, Foster is plainly in the field of securing data. The security that Foster provides is also user-independent (at least) because the user does not select the encryption codes or methods, and a user is not responsible for selecting or managing encryption keys. The security of Foster’s technique does not depend on the user maintaining security in any way. SFI does not, for example, allege that Foster is non-analogous art to the ’609 patent.

9. The ’609 patent and Foster also do not have different contexts (even under SFI’s incorrect view that Foster is limited to streaming live video). The ’609 patent explicitly says it can secure “video” data (an example of Foster’s *data set*) and that the ’609 patent’s “secure parser...may be integrated in any suitable system that involves data in motion (i.e., transfer of data from one location to another),” “includ[ing]...streaming data broadcasts.” EX1001, 68:14-58. Furthermore, as discussed below (§II.C.1, §II.D.1), Foster explicitly discloses embodiments that including storing encrypted content (before and after delivery to customers). Moreover, as I discussed in my First Declaration and further below (§II.C.2, §II.D.2), POSAs had reason for Foster’s processor to store encrypted content for later use (e.g., for video-on-demand delivery).

10. In addition, as even SFI recognizes, one of Foster’s contexts is for use in a conditional access system (“CAS”). Foster, [0005]-[0006], [0016], [0052], [0063]-[0064], FIG. 2. POR, 45. The other one of Foster’s contexts is for a

Digital Rights Management System (“DRMS”). Foster, [0005]-[0006], [0015], [0049]-[0052], [0063]-[0064], FIG. 1. As SFI’s expert Dr. Rubin admitted at deposition, CAS is used “any place that you have permission and **you want to prevent people** who aren’t authorized **from getting something**, you could use conditional access system.” Rubin-Depo., 85, 96-97 (testifying that DRMS “has all the functionality of CAS, but it does other things as well”). *See also infra* §II.C.1, §II.D.1 (discussing further Foster’s CAS and DRMS implementations). Thus, in my opinion, POSAs would disagree that Foster and the 609 are in different contexts or address different problems (let alone “entirely different” as SFI alleges).

11. As I discussed in my First Declaration (§§164-166), Foster describes benefits to attaching an encryption key to the data that that encryption key encrypts. Foster, [0002]-[0007]. Foster explains that doing that provides several benefits. For example, by keeping the encryption key with the *share* of Foster’s data that it encrypted, the risk of applying the wrong key to the wrong piece of share of data is reduced (because the system would not have to “search” for the key, such as in systems where keys are stored and/or delivered separately from the data they encrypt). Foster, [0002]-[0007]. Placing the encryption key with the share of Foster’s data that it encrypts **also** reduces the time it takes to locate the correct key for decrypting a portion of the dataset (i.e., reduces latency), thereby

facilitating “random access” to portions of “protected content.” Foster, [0002]-[0007]. The benefit of reducing latency for random access to content is **not** limited to streaming content, let alone streaming “live” content. Instead, as I explained in my First Declaration (§165), Foster’s technique allows for random access to stored content (e.g., movie) at a particular (“random”) *chunk*: for example, starting a stored movie at any point (say the middle where your favorite scene is), rather than have to start the movie at the beginning every time. Foster, [0002]-[0007]. Foster says attaching an encryption key with the encrypted data helps maintain “synchronization” between the keys and data. Foster, [0002]-[0006].

12. In its POR, SFI argues (and Dr. Rubin opines) that Foster “contemplates th[e] synchronized transmission in the streaming and television contexts.” POR, 54. To the extent SFI is arguing that synchronization for live transmission is the **only** goal of synchronization in Foster, in my opinion POSAs would disagree. Using synchronization to support streaming live content is a benefit in Foster, when Foster’s teachings are applied to streaming live content. But that is just one of the applications Foster discusses. Foster certainly teaches “synchronized transmission to...a receiver.” Foster, [0008]. But in that very same sentence, Foster **also** teaches attaching an encryption key to encrypted data to allow “for synchronized...**storage** by a receiver.” Foster, [0008]. That is, Foster is **explicit** that synchronization can be used to benefit transmission **or storage** of

content (or both in combination). The benefits of synchronization that Foster considers apply to **stored** data as well because, for example, as I explained above, Foster's synchronization supports random access to a movie at a random point in the movie, rather than require starting the movie at the beginning and waiting for the movie to reach the portion you're interested in viewing. The ability to randomly access a portion of movie applies to movies that are stored, whether at a server, an optical-disk storage device (e.g., DVD) or a storage device in a set-top box (among any other storage device from which Foster's content could be stored and retrieved).

13. My opinion that Foster's synchronization is not limited to live-streaming applications is reinforced by Foster repeatedly saying synchronization is useful for either data "transmission to, **or storage by**, a recipient." Foster, Abstract; *see also* [0007]-[0008], [0048], [0064], [0079], claim 11. Indeed, SFI **admits** that Foster "**teaches**...the 'recipient 430 can....**store** the encrypted content.'" POR, 44 (quoting Foster, [0079]) , 51 (similar). Thus, it is readily apparent to POSAs and made clear by multiple disclosures in Foster that Foster includes processing a data for the purposes of securely **storing** the Foster-encrypted data.

14. As demonstrated by the following quote from Foster, Foster is even explicit that synchronization reduces latency for streaming purposes **and** storage-

based purposes, as would be used for Digital Rights Management Systems

(“DRMS”):

[A] recipient will receive or *store* encrypted content along with all information necessary to decrypt the content. This not only prevents the synchronization problems associated with previous systems, but also provides random access of content. As indicated above, both DRMS and CAS have inherent latencies (e.g., delays) in providing random access of content. Specifically, because each content packet transmitted under the present invention is “packaged” with the title key that was used to encrypt it, a recipient can access and decrypt the content stream at any packet.

Foster, [0064].

15. As Foster explains, in Foster’s DRMS implementations, “content is packaged and loaded [i.e., stored] onto a web server” and subsequently “**downloaded to** [i.e., stored at a] personal computer” for on-demand accessing at “rendering time” (i.e., when the user wants to view stored content at some indefinite time after downloading/storing). Foster, [0005], [0049] (explaining how users access “content on media server,” which POSAs understood is *storing* the content before it has been purchased so it could be downloaded by multiple users at various times); FIG. 1 (content is accessed for “streaming **or stored**”).

16. Foster at [0050] says that “the content **can** be delivered through streaming delivery (e.g., if the file is large).” Foster says it “can” be delivered via streaming “**if**” the file is large because a file can also **not** be delivered via streaming (e.g., when it is **not** a large file or whether the storage device has

sufficient memory to store the file). POSAs would understand that what Foster at [0050] is suggesting is that if someone is downloading a large file (e.g., a lengthy, high-definition movie) then it may take a long time to download and store the entire file. Thus, rather than wait for the entire file to be downloaded, a user could start viewing the content via streaming as it is being downloaded.

17. That POSAs would have understood Foster's DRMS embodiments to be for the purposes of **storing** content is also corroborated by SFI's EX2039, which recognizes that DRMS "safeguards content when **at rest** [i.e., stored] in addition to the transport phase," and extends to content "**stored** on a receiving device" and for "offline playback," *i.e.*, not just for live streaming (*i.e.*, "offline"). EX2039, 6-10.

18. Consistent with Foster's DRMS disclosures, Foster also teaches that its payloads can be processed according to the MPEG-standard's so-called "program stream," which Foster says (and POSAs knew) is intended for use "**in fixed storage.**" Foster, [0073]. My opinion that POSAs knew as part of their background knowledge that an MPEG program stream is designed and intended for storage is support by the following corroborating references.

19. In response to SFI's arguments and Dr. Rubin's opinions that Foster is only in the context of synchronization for live-broadcast streaming, I cite as

contrary evidence SFI's EX2014, which acknowledges that the MPEG "[p]rogram streams are **designed for...storage.**" EX2014, 1.

20. In response to SFI's arguments and Dr. Rubin's opinions that Foster is only in the context of synchronization for live-broadcast streaming, I cite as contrary evidence EX1108. EX1108 teaches that **both "[t]he program and transport streams** may be recorded on [i.e., stored on] or played back from a digital video disc (DVD), video tape, magnetic or optical disk drive or other suitable storage device" (EX1108, 1:65-2:31), which are all examples of storage devices in the '609 patent (EX1001, 66:39-45, 66:62-67:11, 70:1-12). EX1108 further discloses that it was known that even with the MPEG transport stream it was conventional that "a prior art video server may interact with a peripheral **such as a storage device** in a television set-top box **to download a transport stream corresponding to a video-on-demand selection.**" EX1108, 1:65-2:31. That is consistent with my opinion (discussed further below) that Foster describes (or is at least suggesting and is consistent with) a video-on-demand (VOD) system, where encrypted content is both stored at the provider (e.g., a cable company and/or movie studio server) **and** at the recipient (e.g., in a cable customer's set-top box). *See also* Foster, [0052] (disclosing that content is "delivered to receiver 28 (e.g., a set-top box)").

21. In response to SFI's arguments and Dr. Rubin's opinions that Foster is only in the context of synchronization for live-broadcast streaming, I cite as contrary evidence EX1109 (an IEEE article from 1995).² EX1109 teaches that the "MPEG-2 System layer describes how elementary streams comprising one or more programmes are multiplexed together to form a single data stream suitable for digital transmission **or storage**." EX1109, 1. EX1109 further taught that "[t]he programme stream ... is **intended for the storage** and retrieval of programme materials from **digital storage media**." EX1109, 2; id., 3-9 (explaining how "data bytes [are] taken sequentially from the original elementary stream" to form a

² The IEEE Xplore website indicates that EX1109 was published in "IEE Colloquium on MPEG-2 - What it is and What it isn't," in connection with a conference on "24 January 1995"; IEEE Xplore also indicates that EX1109 was added to IEEE Xplore on "06 August 2002."

<https://ieeexplore.ieee.org/document/478032>. Based on the above evidence and my review of the indicia appearing on EX1109 and my understanding of the IEEE and IEEE Xplore, it is my opinion that EX1109 was publicly accessible in the prior art by being disseminated or otherwise made available to POSAs exercising reasonable diligence by February 1995, or at least by August 2002.

“PES-packet”), 13 (“The ‘programme stream’ is biased **towards the storage** and replay of a single programme **from a digital storage device....**”).

22. In response to SFI’s arguments and Dr. Rubin’s opinions that Foster is only in the context of synchronization for live-broadcast streaming, I cite as contrary evidence EX1072. EX1072 discloses as background information that content distributors (e.g., cable and satellite companies) had been developing and releasing “[i]nteractive television (iTV)” that provided, among other features, “2) personal video recorders (PVRs) which enable users to record, store, pause, and playback programming using hard-drive storage mechanisms, as opposed to conventional video tape, 3) video on demand (VOD) which enables viewers to order any type of content on demand, such as movies, television shows, special events, sports programs, and other content, and to control its delivery on their receiving devices by pausing, rewinding, or fast-forwarding them.” EX1072, [0002]-[0003]. EX1072 further recognized as background information, that the “content distributor’s access control system, often referred to as a Conditional Access system” would interact with “a user’s set-top box” having “the appropriate middleware software that is compatible with the iTV applications being deployed at the content distributor's head-end.” EX1072, [0004].

23. EX1072 further discusses that as background information that it was “desirable to have methods and systems that allow **a service provider to devise,**

deploy and manage ... video-on-demand service packages.” EX1072, [0011].

In EX1072’s example, prior-art system, included numerous places where content would be stored – both before delivery and after delivery to customers:

[0042] ... The system operates with **a built-in storage device in the client device for storing the programs, for instance, a hard disk drive or any other storage media or by storing programs on a server at the head-end**. Exemplary personal video recorders are designed by companies such as TiVo, Inc. and manufactured by companies such as Sony Corporation and U.S. Phillips, Inc.

[0043] Video on demand (VOD) 345 enables the subscribers to select a particular content type from a list of **available content stored on a plurality of servers at the head-end to be viewed by the subscribers**. The selected content type could be television programs, movies, events, sporting events, music or **some other form of stored media** or real-time encoded broadcast. The VOD servers send the requested content to the subscriber over a delivery mechanism such as cable, satellite, or other transmission infrastructure. The client device may have **some storage media in which the delivered content can be stored** and viewed by the users. The user may also be provided with the transport control functions such as controlling the delivered video. This may include pause, fast forward, and so forth. **The transport control functions are typically supported at the headend where the content is located/stored**.

24. Accordingly, that Foster’s teachings include storage-based embodiments is readily apparent from Foster’s text, as explicitly stated in Foster and POSAs would independently understand from their background knowledge (corroborated above) of the contexts (e.g., CAS and DRMS) and encoding techniques (e.g., for program streams and transport streams) explicitly discussed in Foster.

25. Finally, even in the implementations of Foster that are for streaming content for use in a conditional access system (“CAS”), those systems do not only deliver “live” broadcasts. For example, POSAs knew that CAS is used in video-on-demand (VOD) systems, where “content **is stored pre-encrypted** on hard drives in the VOD server.” EX1110, [0008]. POSAs similarly knew that the content delivered by VOD was commonly stored again at a “client device[’s]” or in “**storage** media in which the **delivered content can be stored**” (EX1072, [0043]).

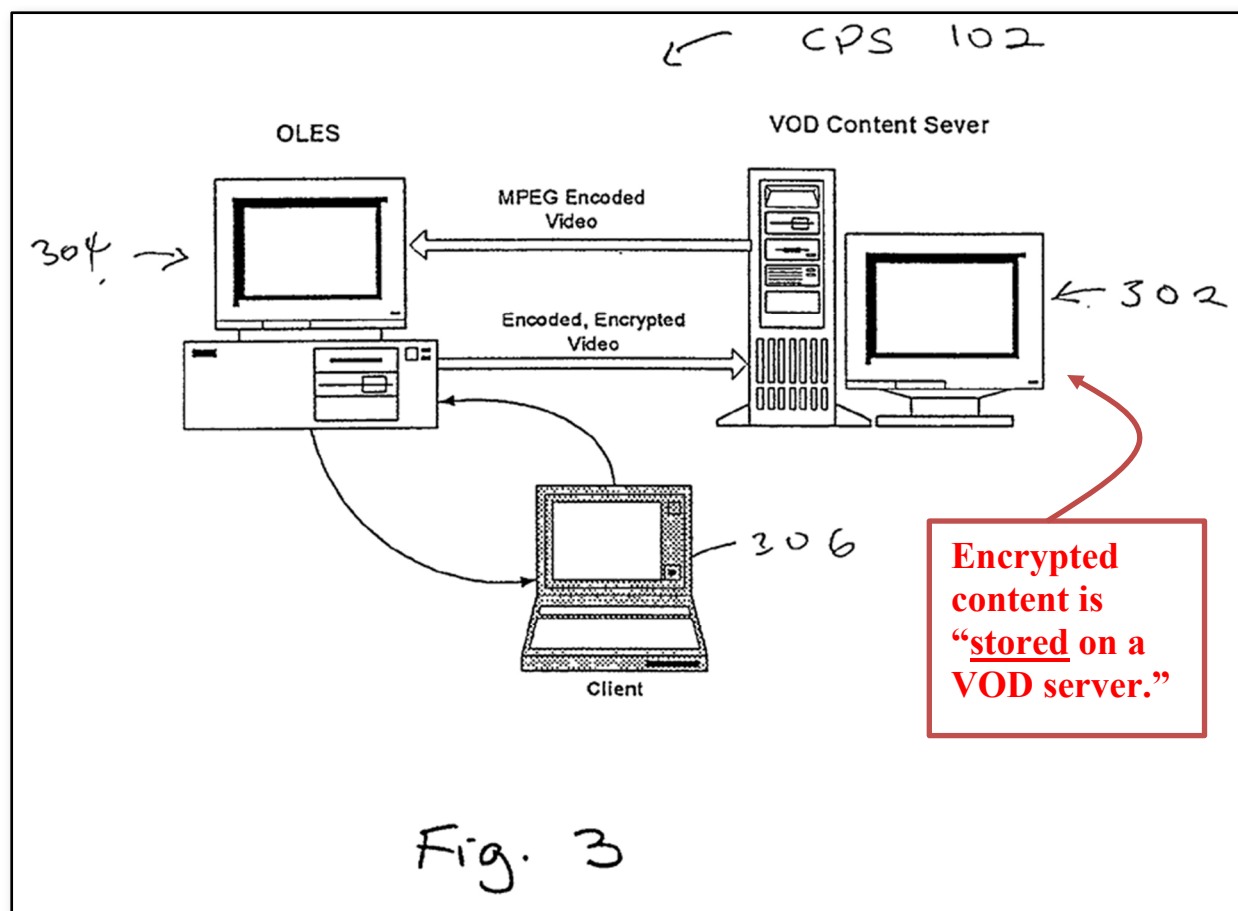
26. My opinion that POSAs understood that Foster’s CAS embodiments were broader than “live” streaming and also relevant to the *storing* of encrypted content is corroborated by the following references (along with the references I discussed above).

27. Hayhurst (EX1116) – which I discuss in depth in my declarations for the ’854 Patent – recognizes that in “deliver-on-demand,” such as for “cable television (CATV) system (or for other content provider[s]),” it was desirable for

the end-user consumer (e.g., the user that has purchased and received an encrypted movie file), to “assemble[the segments] into the complete media file (movie) for display, **or stored for later display**, on the requesting subscriber’s unit.”

Hayhurst, [0004]-[0006].

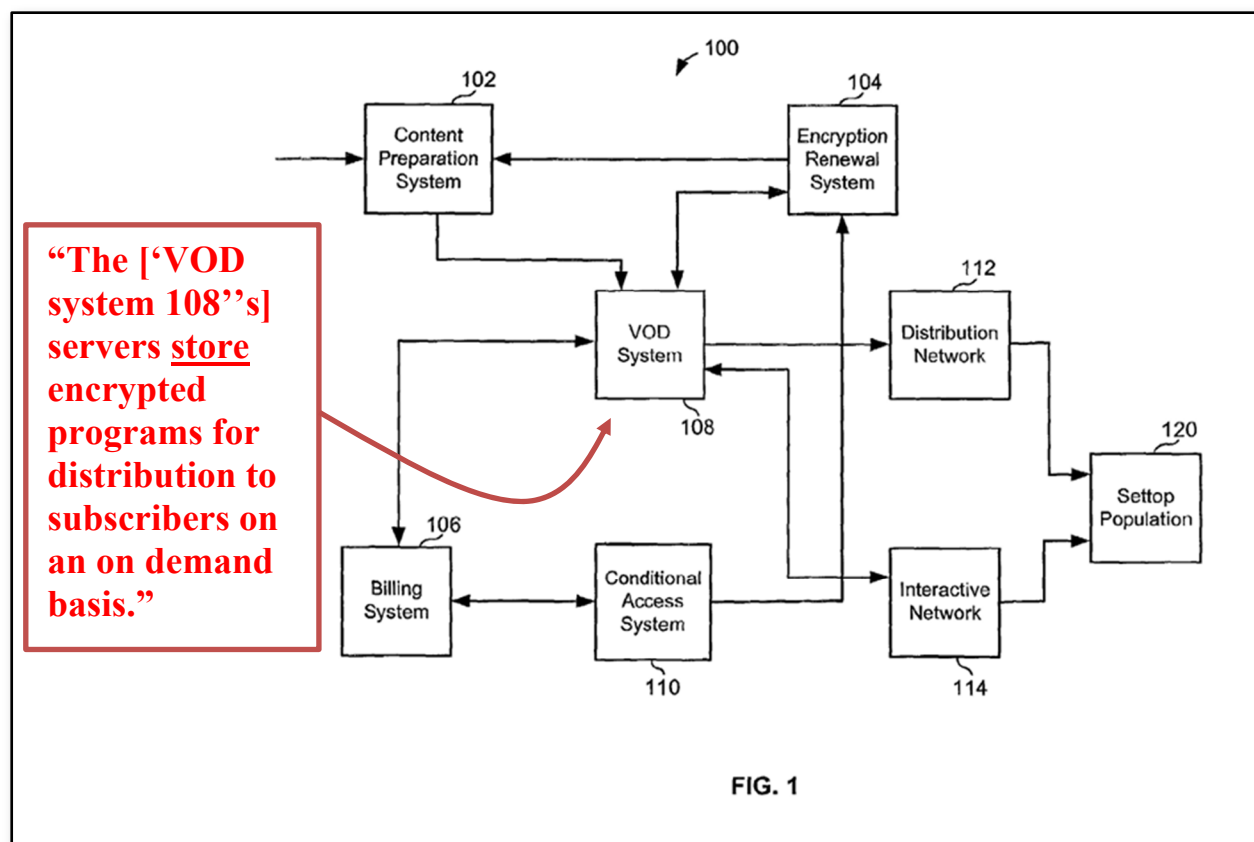
28. In response to SFI’s arguments and Dr. Rubin’s opinions that Foster’s CAS implementation would only be for use in streaming live television, I cite EX1073. EX1073 at [0001]-[0004] corroborates by opinion that POSAs knew that “conditional access” systems were used for VOD system. EX1073 at [0043] further corroborates that it was known that VOD systems had “pre-encrypt[ed]” content and “**stores** the content for forwarding...when authorized” for delivery to a storage device. *See also* EX1073, [0098] (“The encoded file is encrypted and verified prior to writing the encrypted material to VOD content server 302..... A streaming mode of pre-encryption is also possible in which content is ‘streamed’ from a source of raw content such as a video (possibly analog tape) through an MPEG encoder, sent to the OLES to ***perform encryption and finally stored on a VOD server.***”), FIG. 3 (annotated below).



29. In response to SFI's arguments and Dr. Rubin's opinions that Foster's CAS implementation would only be for use in streaming live television, I cite EX1097. EX1097 at 2:31-53 describes as background knowledge that using "real-time encryption" in VOD system is problematic and "poses much greater cost and space issues." Thus, rather than use real-time encryption, it was known to implement content-delivery systems (like Foster and Hayhurst) that "that **encrypts content offline (typically before the content is requested by the user)** before it is distributed to point-to-point systems such as cable systems. The system allows content to be encrypted once, **at a centralized facility**, and to be useable at

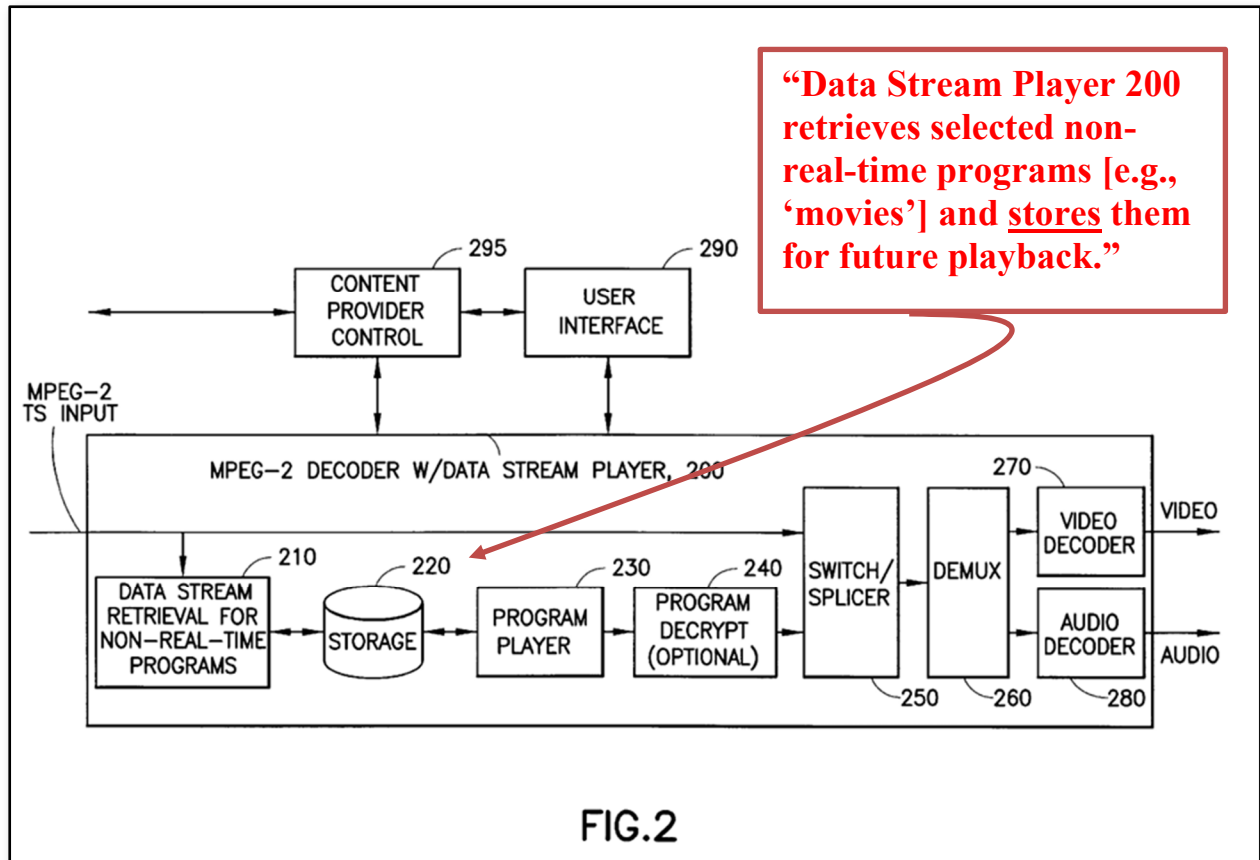
different point-to-point systems.” EX1097, 2:31-53, FIG. 1 (annotated below).

That is, the content is encrypted and **stored** at the “centralized facility” so it could be delivered (again and again if popular content). Thus, EX1097 is also consistent with my opinion that POSAs would have wanted to pursue Foster-Hayhurst for the various benefits that POSAs would have recognized, such as improving the aggregate speed of content delivery and fault tolerance. First Declaration, ¶¶182-185.

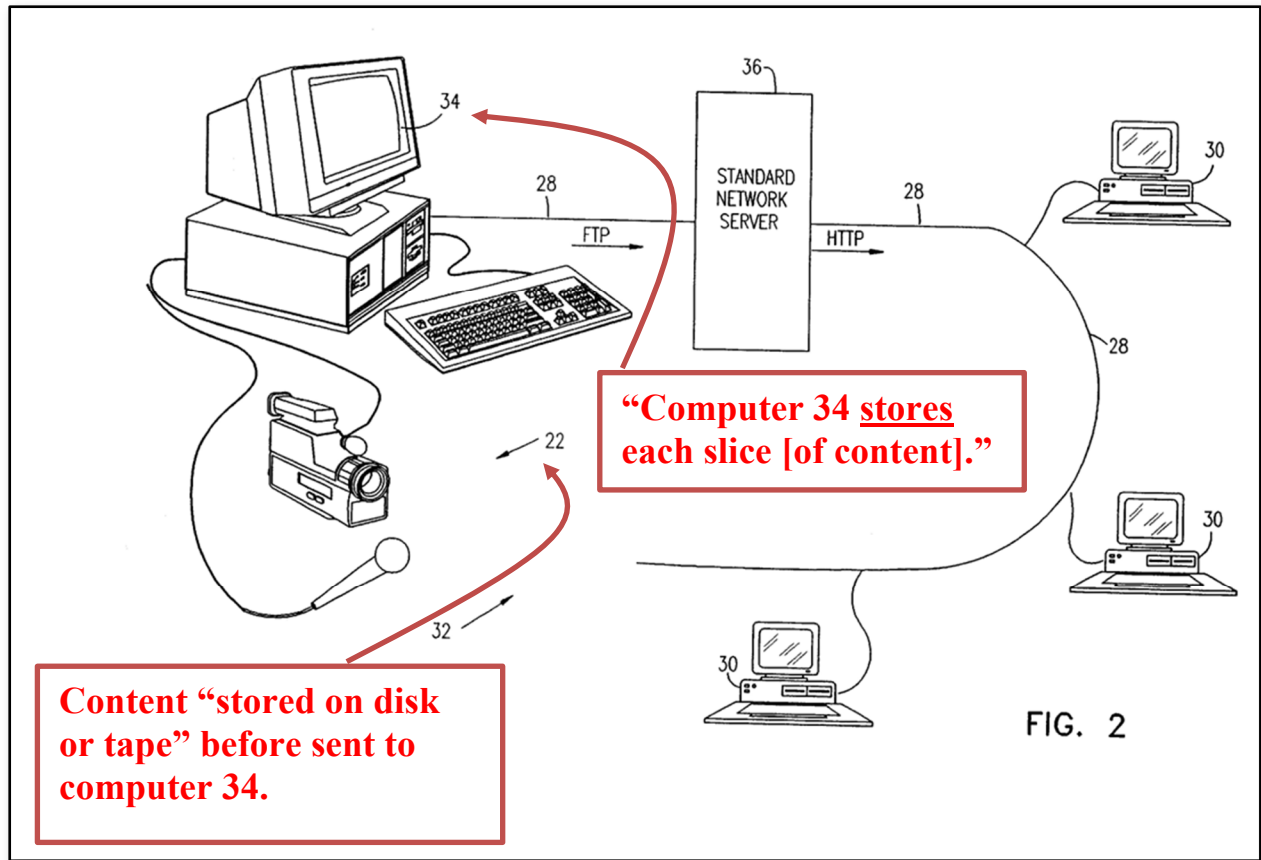


30. In response to SFI’s arguments and Dr. Rubin’s opinions that Foster’s CAS implementation would only be for use in streaming live television, I cite EX1075. EX1075 at 1:51-67 teaches that it was known to implement the “MPEG-

2 transport” stream in a “system [that] should allow the programs and other content to be provided *on a non-real-time basis for later use by a recipient*. Such content should be able to be easily received, *recorded on a digital VCR or otherwise stored*, and played back at the user’s convenience.” EX1075 at 1:51-67 further confirms that it was desirable to have “the delivery of *pre-encrypted content* ... in *non-real-time*.” *See also* EX1075, FIG. 2 (annotated below). Thus, EX1075 further corroborates my opinion that POSAs not only understood Foster’s program stream to be beneficially and primarily directed to storage, but that it was beneficial to *non-real-time* storage and delivery of encrypted content (i.e., *not* live streaming).

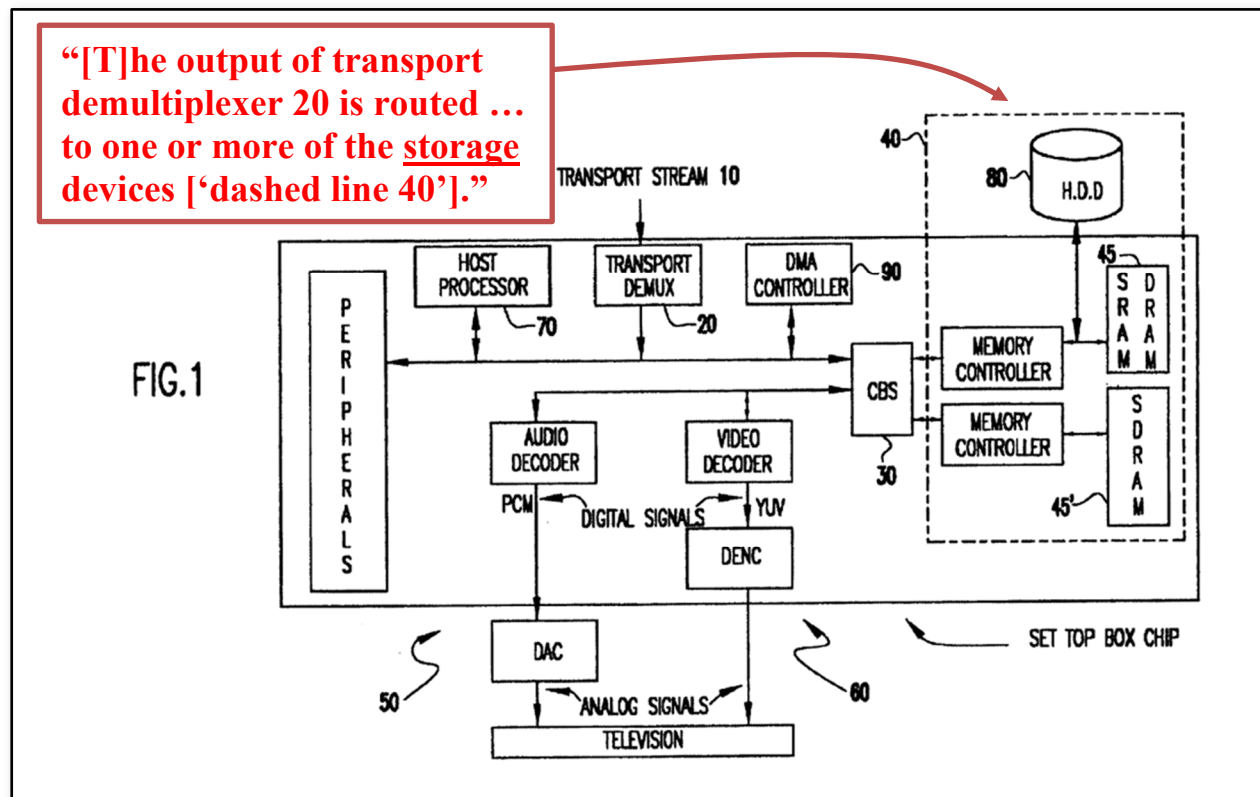


31. In response to SFI’s arguments and Dr. Rubin’s opinions that Foster’s CAS implementation would only be for use in streaming live television, I cite EX1076. EX1076 at 9:53-10:5 discloses a prior art system where even “Broadcast data are [] input to the computer, for example, from input devices 22, or from a video, audio or animation sequence *stored on disk or tape.*” That is, even “live” broadcasting of content is rarely real-time, but instead often retrieved from a *stored device* that stores the content for some later use.



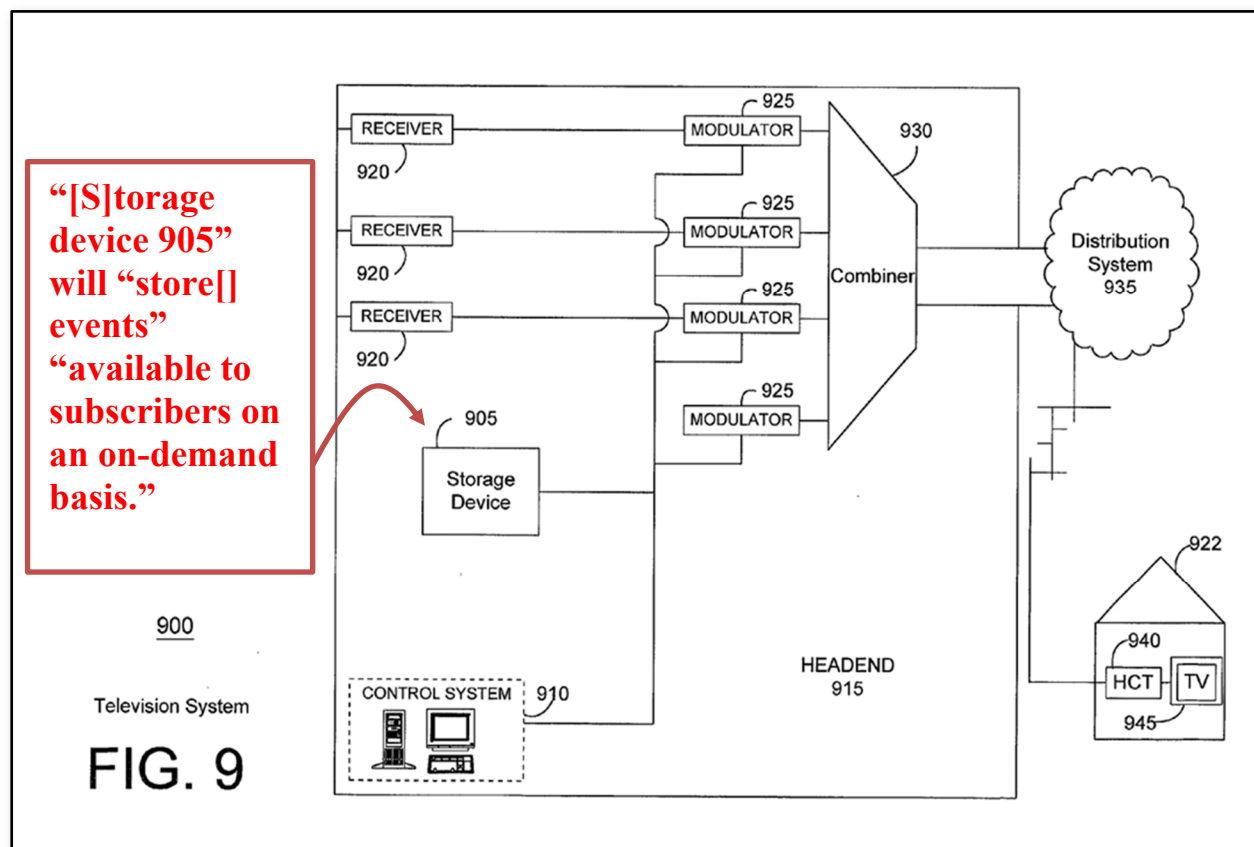
32. In response to SFI's arguments and Dr. Rubin's opinions that Foster's CAS implementation would only be for use in streaming live television, I cite EX1077. EX1077 (which is another patent by several of the Foster inventors), which discloses that it was known in the prior art to have "pay-per-view movies" that were viewed "at a time convenient" to the user. EX1077, 1:12-21. In those system, users typically had a "set-top box (STB)" (as some of Foster's recipients do, *see* [0040]), that was expected to have "substantial storage within the STB" and/or paired with "further storage on an onboard mass storage device such as a hard disk drive or compact disk recorder/playback device." EX1077, 1:48-65.

EX1077 at 3:56-5:8 goes on to describe processing of an MPEG “packetized elementary stream” (as in Foster) for “stor[ing] to mass storage.”



33. In response to SFI’s arguments and Dr. Rubin’s opinions that Foster’s CAS implementation would only be for use in streaming live television, I cite EX1092. EX1092’s Figure 9 (annotated below) illustrates “a storage device 905, such as a video-on-demand server.” EX1092, [0079]. “A control system 910 within the headend 915 is **programmed to store all the streams** included within popular events that are received from one of the receivers 920” (where the receivers are receiving content from content providers). That is, even in cable / satellite company systems, a conventional video-on-demand server for CAS (like

those discussed in Foster) will have a processor that instructs storage devices to store content for later downloading and storage to another storage device.



Thus, it is clear from Foster’s disclosure (and reinforced POSAs’ background knowledge that I corroborated above) that Foster’s technique was designed for use in CAS-based systems that included the storage of content both before and after delivery of the encrypted content to end-user recipients (both for streaming and for non-streaming applications).

34. Foster’s compatibility with CAS-based and/or VOD-based system also corroborates the opinions expressed in my First Declaration that POSAs had

motivation to implement Foster where content encrypted by Foster's method was stored in the memory of storage devices coupled to Foster's processor based on the processor executing code for *storing* on at least one of those storage devices *at least one data chunk of the plurality of data chunks with data indicative of at least one of the distinct encryption keys*. See First Declaration, ¶¶176-177, ¶¶200-207; *see also infra* §II.C.1, §II.D.2.

35. Thus, as I explained in my First Declaration (¶¶206-213) with corroboration, POSAs had motivation to pursue the obvious modifications of Foster that I identified for purposes of storing encrypted content for use in both streaming applications (e.g., streamed VOD) and non-streaming applications (e.g., VOD for downloading and storage).

B. Response to Argument that Foster's Encrypted Payloads Are Not *Data Chunks*

36. Claim 1 requires “*generating a plurality of data chunks based on a data set*” and then encrypting those *data chunks with a respective second key*. SFI acknowledges the Petition (and similarly my First Declaration) “map[ped] the claimed ‘plurality of data chunks to Foster’s’ “MPEG packet payloads, [*i.e.*, which are within Foster’s content units],” and “the claimed ‘data set’ to Foster’s [elementary digital] content.” POR, 34-35 (citing, *inter alia*, Petition, 65-66); *see* First Declaration, ¶¶184-189.

37. Based on my review of SFI's POR and Dr. Rubin's declaration, it does not appear that SFI is disputing that Foster's content (e.g., Foster's elementary content unit) meets the claimed *data set*.

38. In its POR, SFI argues that Foster's MPEG payloads (which is comprised within Foster's content units) are not and cannot be *data chunks*. POR, 34-35. According to SFI, a payload of a data set processed consistent with (or in ways like) the MPEG standard cannot be a *data chunks*. In my opinion, as I explain further below, SFI's arguments are incorrect and there nothing in the intrinsic record of the patent to support SFI's narrow interpretations of *data chunk*.

39. SFI argues that the Petition "is conclusory" because it did not explain "what 'data chunks' are or why [Foster's] packet 'payloads' would be deemed a chunk." POR, 66. I note that neither party has construed *data chunk* as its meaning would be readily apparent to POSAs from the claim language.

40. In my opinion, the plain language of claim 1 provides that a *data set* is the set of data to be securely stored and a *data chunk* is a portion (less than all) of that data set. I note that at deposition Dr. Rubin testified that "the share is the thing that something gets split into" (Rubin-Depo., 17) and that the terms "share" and "chunk" just "refer[] to the pieces that have been split off from the original data set" (Rubin-Depo., 157-158). Each of Foster's payload's is exactly that: it

comprises data split off from the *data set* being secured, and thus is a *data chunk*, as I discussed in my First Declaration (¶¶184-189).

41. In my opinion, POSAs would understand that the specification places no limit on the relationship between a *data set* and a *share*, so long as it is a subset of the *data set*. The specification has multiple disclosures that inform POSAs that the pieces of data that are split from the data set (i.e., *data chunks* in the '609 patent's claims) can be of any size of data from the data set or can each be the same size or different sizes of data from the data set. For example, the patent says POSAs "will recognize from the disclosure herein, a wide number of alternatives for the data splitting process," and that "data may be split **into any desired, selected, predetermined, or randomly assigned size unit.**" EX1001, 21:9-25. The patent similarly says that a "data unit sizes **may be...all of the same size**, [or] a combination of sizes...." EX1001, 21:30-34. Elsewhere, the patent says "[i]t is readily apparent to those of ordinary skill in the art that **the split data unit sizes may be a wide variety of data unit sizes** or patterns of sizes or combinations of sizes." EX1001, 21:27-31. Similarly, the patent says "[i]t will be understood that any other suitable parsing and splitting approach may be used to generate portions of data from an original data set in accordance with the present invention." EX1001, 74:56-75:10. Thus, the patent explicitly (and repeatedly) allows for

shares (in the '609's claims *data chunks*) to take any form or size of data so long as they are a subset of data split from the *data set*.

42. In its POR, SFI suggests a *share* must be cryptographically generated, e.g., the originally *data set* must be randomly parsed/split, thereby making the data within the *data chunk* less susceptible to “attack.” POR, 39-40 (quoting EX2033, “Rubin-Decl.”, ¶¶100-102 (citing EX1001, 56:44-58)). If SFI is arguing that the claimed *data chunks* must be randomly split from the *data set* or cryptographically split from the *data set*, then POSAs would find that that construction is inconsistent with the plain meaning of the claims or the patent's specification.

43. As I discussed above in this section, the plain meaning of claim 1 does not restrict the manner by which *data chunks* are generated from the *data set* beyond each *data chunk* that is split from the *data set* must comprise less than the entirety of the *data set*.

44. Similarly, the patent's specification says “data units may be distributed into one or more shares” in several different ways, including “**fixed** or predetermined data unit size...or a randomly generated data unit size,” as an **alternative**. EX1001, 21:33-37. That is, having randomly generated data unit size is an **option** but it is not required. As the language I've just quoted shows, the patent is explicit that each share (in the 609's claims *data chunk*) can be a “fixed or predetermined data unit size.” That means, contrary to SFI's arguments, the patent

does **not** require “splitting the data into shares” to “help[] secure the data.” POR, 69. Instead, the patent discloses that the sizing of the splitting can be done to provide no additional security.

45. (I note, however, that one of my obviousness opinions for Ground 2 – Orsini – demonstrated that it would have been obvious to cryptographically adjust the sizes of Orsini’s *data chunks* to further secure the data set based. First Declaration, ¶¶116-119 (citing Orsini, [0281] and additional corroboration)); *see also* Orsini, [0292] (which include the same disclosure as EX1001, 56:47-61).

46. The dependent claims of the ’609 patent are further evidence that SFI is incorrect in its attempt to narrow the scope of claim 1’s *data chunks* to provide protection by being cryptographically split from the *data set* (as opposed to being some portion split from the data set that is less than the entirety of the *data set*). I note that dependent claims 4-6 limit claim 1’s *data chunks* to ones that are generated using cryptographic techniques like a “random technique.” I understand that a dependent claim is narrower than the independent claim from which it depends. Thus, the fact dependent claims from claim 1 require cryptographic techniques for generating the *data chunks*, but claim 1 does not have any such limitation about the splitting in and of itself providing security, further corroborates my opinion that POSAs understood the independent claim is broader

and is not limited to generating *data chunk* using particular techniques that might further increase security of the *data chunks*.

47. In its reply, SFI does not dispute that Foster’s payloads (content units) each comprise less than all of the data in Foster’s content, *i.e.*, *data set*. In my opinion, POSAs understand that that is all that the claims require. The fact that “Foster never refers to its packet payloads as ‘data chunks’” (POR, 37) is irrelevant because a POSA would understand that Foster’s payloads (content units) meet the claimed *data chunks* even if Foster did not label them “shares” or “data chunks.” SFI does not dispute that Foster’s payloads (in Foster’s content units) each comprise less than all of the corresponding content file (*data set*). That is all the claimed *data chunks* require.

48. For (at least) the above reasons, in my opinion POSAs would disagree with SFI that it would have mattered to their understanding of how Foster includes *data chunks* even if it were true that “Foster’s packet payloads [or Hayhurst’s segments] exist not to protect data.” POR, 69 (quoting EX2033, ¶¶149-151).

49. That said, the POR at one point suggests that in the ’609 patent it is merely the act of “splitting the data set into shares (*i.e.*, chunks) [that] help[s] secure the data” (POR, 39). That is, SFI is saying that just by splitting the data set into *data chunks* security has been increased. If that is SFI’s argument, then Foster clearly provides that same security based on generating its payloads. That is

at least because in Foster each *data chunk* can be separately encrypted with its own encryption key, which Foster says “provides enhanced security preventing an entire content stream from being accessed with a single key.” Foster, [0004]; Foster, [0071] (“In addition, because each content packet can be encrypted with a different title key, a party who illicitly obtains a title key cannot access the entire stream. In contrast, the pirated key is only good for decrypting the content packet to which it was attached.”). That supports my opinion that generating Foster’s packet payloads from Foster’s *data set* is **done** to protect data, not merely as a function of the MPEG standard, because Foster splits the data set into *data chunks* for the purpose of protecting the individual pieces of data and overall *data set*. Foster individually encrypts each *data chunk* (packet payload) with an encryption key (potentially a different encryption key). First Declaration, ¶¶164-171, ¶¶174-175, ¶¶184-187, ¶¶194-195; *see, e.g.*, Foster, [0071], FIG. 8. Contrary to SFI, Foster’s splitting and encryption is **not** “just a by-product of a pre-existing standard.” POR, 37-38. The MPEG standard, for example, does not require encrypting each of the packet payloads (let alone encrypting them with respective, different encryption keys as taught for one embodiment of Foster). It is Foster’s encryption technique, not a requirement of the MPEG standard, that encrypts each of the individual *data chunks* for the purpose of protecting the *data set*.

50. I note that SFI has identified several exhibits, at least several of which are not prior art “literature,” that allegedly teach that a “payload” or “packet” can be “divided into ‘chunks’ or ‘shares.’” POR, 37-39 (citing EX2012-EX2013, EX2041-EX2043, EX2036-EX2038). I have reviewed these exhibits. They generally concern generic “packets” (e.g., an IP packet), and are not about packets produced according to the MPEG standard, let alone according to Foster’s disclosures.³

51. In my opinion, the exhibits SFI cites are not inconsistent with my opinion that POSAs’ would have understood that Foster meets the claimed *data*

³ I note too that Foster itself is not limited to using MPEG data, as “Content” is defined as “**any** data such as digital image, sound or binary data deliverable from a source to a recipient.” Foster, [0037]. Thus, Foster (let alone obvious implementations of Foster) would not be limited to dividing its *data set* into MPEG “packets” (or any other packet). Regardless, as I discuss in this section of my reply declaration, I do not agree and a POSA would not agree that a packet of MPEG-processed data cannot be a *data chunk*; and a POSA would find that Foster’s processing in particular generates (and had reason to generate) *data chunks* for the reasons I discussed in my First Declaration (¶¶184-193) and here in response to SFI’s and Dr. Rubin’s arguments.

chunks. As I discussed above, neither the challenged claims nor the patent specification preclude a packet from either (i) being a single *data chunk* or (ii) being divisible into multiple *data chunks*. The issue is whether the data being carried by that packet comprises data from the *data set* that has been generated from a *data set* and comprises less data than the entire *data set* (which the patent claims call a *data chunk* of that *data set*). Foster’s packet payloads are each less than an entire *data set*, as I demonstrated in my First Declaration (¶¶184-187) and discuss above in response to SFI’s POR and Dr. Rubin’s declaration.

52. Moreover, other literature demonstrates that POSAs knew that even something referred to as a “packet” could also be called a “chunk”/ “data chunk” of a data set if (as in Foster and obvious implementations thereof) the packet comprises a less-than-entire portion of the data set.

53. In response to SFI’s and Dr. Rubin’s argument regarding POSAs’ understanding of Foster’s content units, I cite EX1082, which unlike SFI’s exhibits **is** in the context MPEG-encoded data.⁴ EX1082 says that with MPEG, each

⁴ I note that a document sharing the same author, title, and publisher as EX1082 was cited on page 2 of U.S. Patent 7,898,951 (filed Aug. 6, 2004; issued Mar. 1, 2011). See <http://bit.ly/49OIc3o> (PDF of US7,898,951 on USPTO’s website). The ’951 patent lists as a reference titled “Mike Knee, ‘MPEG Video’” and indicates it

packet's "payload [] is a **chunk** of a particular elementary stream." EX1082, 8.

The "elementary [content] stream" is an example *data set* in Foster. Foster, [0004], [0008], [0037], FIG. 8; First Declaration, ¶174, ¶¶184-187.

54. And even as to generic computer "packets," prior art recognizes a "packet" can be a chunk (a "share") of a dataset. For example, EX1083 at 7:32-40 gives an example of "[a] packet [that] is a chunk of data provided by the

was published "Mar. 2002" by "Snell & Wilcox," and was available from <http://www.bretl.com/mpeghtml/codecdial.HTM>. I have also reviewed the Internet Archive's Wayback Machine and identified archived copies of Snell & Wilcox's website, including an archive from May 15, 2006 of a "White Papers" web page. <https://bit.ly/4fuXZWb>. That archived web page provides a link to download "MPEG Video: Mike Knee," which links to a November 9, 2006 archived PDF of EX1082. <https://bit.ly/3ZRVAiu>. See also the following link to a 2006 archived copy of Snell & Wilcox's home page, which provides a link to a "Knowledge Center" web page that links to the "White Papers" web page I identified above. <https://bit.ly/4iPP58B>. See also <https://bit.ly/49P0xxe> (link to IEEE Xplore's page for an article added to IEEE Xplore on January 3, 2006, which lists "Mike Knee, 'MPEG Video,' Snell and Wilcox" as a reference cited in that article, which I have confirmed by reviewing a copy of that January 2006 article).

application program.” Thus, EX1084 contradicts SFI’s argument that something called a “packet” cannot also be “chunk or data.”

55. Similarly, EX1084 at [0026] teaches that an “RTP **packet**” can be “**A chunk or portion of multimedia data** as sent over an IP network...” Thus, EX1084 contradicts SFI’s argument that something called a “packet” cannot also be “chunk or portion of” a data set.

56. As another example, EX1085 (at 4:7-25) describes a system that “disassembling a data set into a plurality of data blocks...transforming each of the data blocks into L **data chunks**,” where “[a] size of **each data chunk can correspond to packet size**” (i.e., there is only one chunk-per-packet). Thus, EX1085 contradicts SFI’s argument that something called a “packet” cannot also be “data chunk” of a data set or that a packet is only divisible into data chunks rather than itself being the data chunk.

57. Thus, the above references corroborate my opinion that SFI is incorrect in its argument that POSAs would understand that packets cannot be *data chunks* merely because packets could, in some systems, **also** be “divided into data chunks” (POR, 37-39). The above references demonstrate that POSAs would disagree, as do I.

C. Response to Argument that Foster Does not Disclose or Render Obvious the Processor Executing Code To Store at least one Data Chunk on a Memory Coupled to the Processor ([1A], [1F])

1. It is still my opinion that Foster Discloses that its Processor is Executing Code to Store at least one Data Chunk on a Memory Coupled to Foster's Processor

58. Claim 1 recites “*executing code by a processor to perform*” ([1A]) “*storing, in a memory coupled to the processor, at least one data chunk...on at least one storage device*” ([1F]). SFI argues Petitioner “does not attempt to show that Foster’s CPU 406, executes any code **relating to storing content** units on recipient 430 or instructing recipient 430 to store content.” POR, 43. In my opinion, POSAs would not agree that Foster does not meet limitations [1A] and [1F], which I demonstrated in my First Declaration (¶¶176-177, ¶¶200-214).

59. SFI does not dispute that, as Petitioner argued, Foster’s processor “execut[es] ‘a computer program’” to implement Foster’s encryption methods. Petition, 62 (quoting Foster, [0074]-[0075]). As I discussed in my First Declaration (¶¶200-214), Foster’s processor executes code that is related to storing content units on storage devices, which the POR (at 43) says meets [1A]/[1F]. Those storage devices include servers coupled to Foster’s processor that deliver stored encrypted content to Foster’s recipients, which in turn include storage devices coupled to Foster’s processor that also store the content.

60. I note too that Dr. Rubin testified at deposition that the claimed *code* “has to be code executed on the processor **that results in storing** something on – **in memory that’s coupled to the processor.**” Rubin-Depo, 189-191. *Infra* §II.D (where I discuss the *coupled to* limitation). As I discussed in my First Declaration (¶¶200-214) and further below in response to SFI’s and Dr. Rubin’s arguments, Foster’s processor executes code that results in storing Foster-encrypted data in a memory that is coupled to Foster’s processor.

61. SFI argues Foster’s processor is “agnostic as to what the recipient [] does with the content units after transmission,” and fails to “instruct the recipient [] to store or perform some action related to storing.” POR, 43-44. This is a new claim construction by SFI (EX1060) and SFI and Dr. Rubin identify no intrinsic evidence requiring the processor to send instructions to a storage device. Contrary to SFI’s arguments otherwise, Limitation [1F] only requires executing code “*to perform storing*” in a memory coupled to the processor. Foster’s processor does that because it executes code that encrypts content that is stored in the memory of storage devices that are coupled to Foster’s processor (e.g., the servers that the processor instructs to store content for later storing by other recipient storage devices that are coupled to Foster’s processor).

62. Regardless, even if the Board were to adopt SFI's new, narrow claim construction, in my opinion, POSAs would still find that Foster anticipates claim 1 in several ways.

63. The first way that Foster anticipates under SFI's new, narrowing construction is based on Foster's processor executing code to store encrypted content in a local memory device that is *coupled to* the processor. I first note that it is my understanding that at institution the Board disagreed with SFI's pre-institution argument that the claimed storage device must be for long-term storage. DI (Paper 16), 18. SFI did not make that same long-term storage argument in its POR, and thus SFI's POR is no longer arguing that the claimed *storage device* must be for long-term storage. As part of executing instructions to encrypt the *data set*, Foster's processor is executing instructions to store encrypted *data chunks* on storage-device memory (e.g., RAM) that is directly *coupled to* the processor via Foster's bus. *See* Foster, FIG. 16 (discussed below).

64. My opinion that a conventional CPU / processor like Foster's will execute code that instructs a memory device *coupled to* the processor to store the data being processed by the processor, is corroborated by EX1117. EX1117 describes as background information the following aspects of a "conventional computer":

The basic components of a conventional computer typically include one or more processors for processing program instructions; a temporary memory device (such as random access memory, or RAM) for storing program data and instructions that are being processed by the processors; a permanent memory device (such as a magnetic disk) for storing the computer's operating system, programs and **data; and various input/output devices (such as a monitor, a keyboard, network adapters and others) for facilitating the necessary interactions between the computer, human users and other technical systems.**

EX1117, [0003]. Thus, EX1117 corroborates my opinion that, as part of executing Foster's methods, Foster's processor will execute instructions to encrypt data and store the data being processed (which will include at least one Foster-encrypted *data chunk*) in a "memory [i.e., storage device]" like RAM. EX1117, [0002]. EX1117 further corroborates that it would have been conventional for Foster's processor to then also execute instructions to store data after processing it in a "permanent memory device," like a "magnetic disk[, e.g., a DVD disk] for storing ... data." EX1117, [0002]. EX1117 further notes as background information that in a conventional computer the "**components in the computer are interconnected by various means**, including a direct channel, a system bus, and/or one or more peripheral busses. Typically, **the processor interacts with the temporary memory device and the input/output devices.**" EX1117, [0005]. In other words,

consistent with my opinions regarding Foster's computer, EX1117 corroborates that a conventional processor is *coupled to* various storage devices, including a memory device like RAM for storing data as part of processing the data, a permanent storage device for storing all data after it has been processed, and possibly additional storage devices *coupled to* the processor via input/output devices (like the communication interfaces in Foster's computer).

65. That is, as Foster's processor executes code to secure data, the processor is executing instructions to store encrypted data (and thus *at least one data chunk*) in memory, in particular Foster's memory 402 that is shown in Foster's Figure 16, which I have annotated below. Foster explains that Figure 16's computer system – which, among other things, executes Foster's encryption method – “comprises memory 402,” which “may comprise any known type of data storage and/or transmission media, including management media, optical media, random access memory (RAM), read-only memory (ROM), a data cache, a data object, etc.” Foster, [0075]. Foster further notes that “memory 402 may reside at a single physical location, comprising one or more types of data storage, or be distributed across a plurality of physical systems in various forms.” Foster, [0075]. Foster explains that “[b]us 410 provides a communication link between each of the components in the computer system 400 and likewise may comprise any known type of transmission link.” Foster, [0076]. As Figure 16 (annotated below)

illustrates, one link that bus 410 provides is between the processor and memory. Foster further explains that “[s]tored in memory 402 is attachment system 412, which includes parsing system 414, encryption system 416, key system 418, designation system 420, combination system 422 and transmission system 424.” Foster, [0078].

66. Thus, all instructions to process and store data, and all data processed according to those instructions (*i.e.*, at least one encrypted *data chunks*), are stored in memory 402 (among other storage devices that I discuss below). That is, all data processed by Foster’s encryption system (including at least one encrypted *data chunk*) is at least initially stored in memory 402, which as illustrated below is directly *coupled to* Foster’s processor via bus 410. And then, as I discuss below, the processor executes further instructions to cause the encrypted data to be stored in other storage devices *coupled to* the processor.

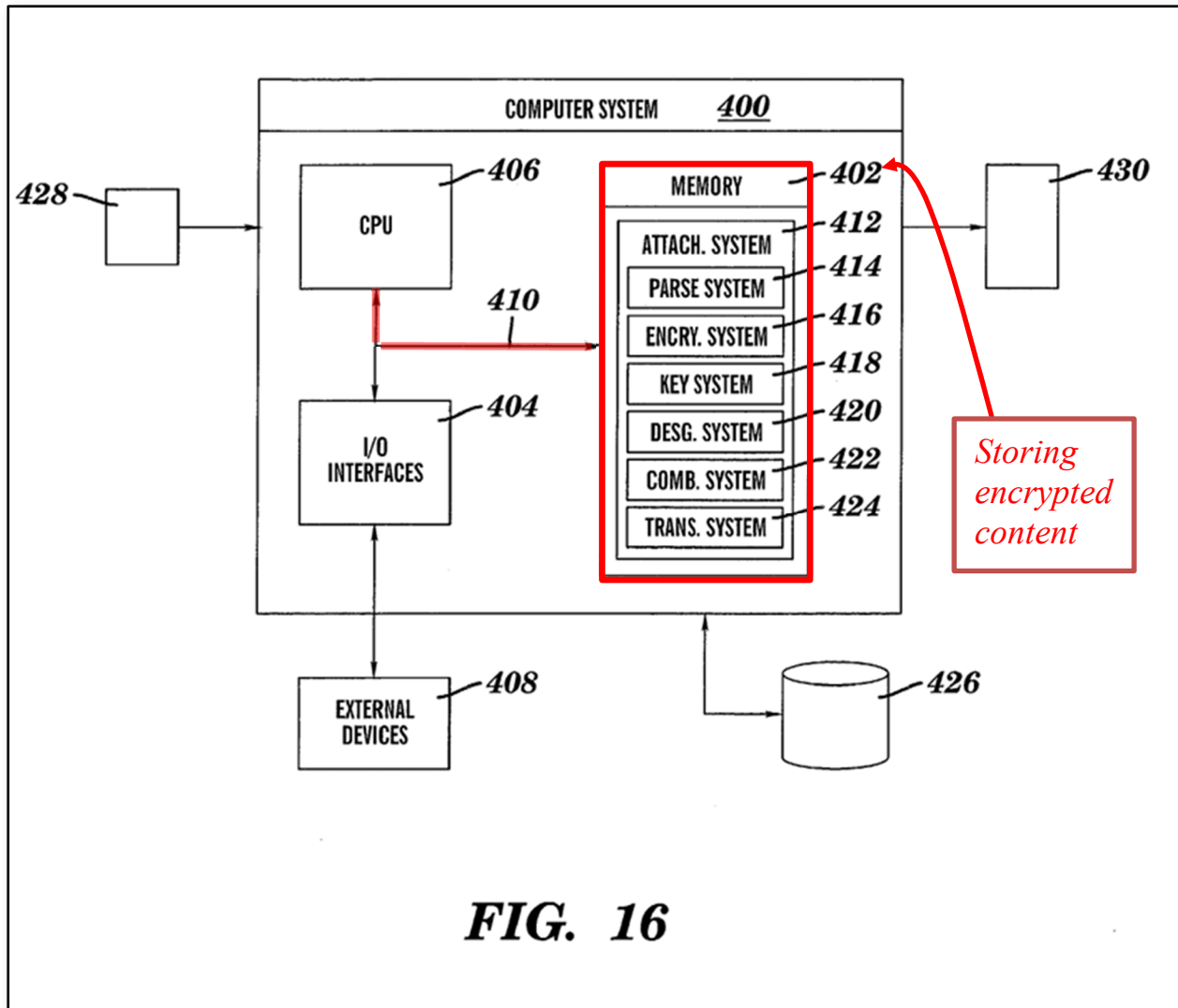


FIG. 16

67. For at least the above reason, POSAs understand that Foster discloses limitations [1A] and [1F] under SFI's view of the claim scope that requires the processor to instruct the memory *coupled to* the processor to store a *data chunk*.⁵

⁵ For clarity, I have not been asked to consider whether SFI's new, narrow claim construction is correct (i.e., I have not been asked to opine on whether claim 1

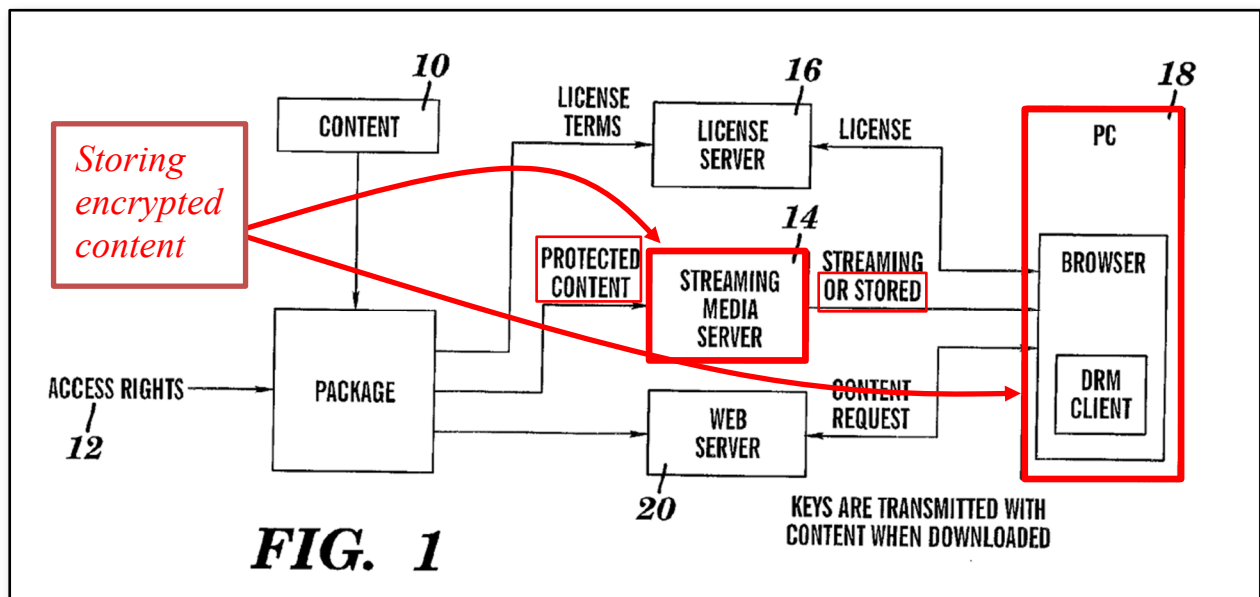
68. Foster discloses [1A] and [1F] in additional ways, under SFI's new, narrow claim constructions, that are also consistent with my opinions in my First Declaration. As I discussed above (§§II.A-B), Foster's processor executes code to encrypt content for DRMS- or CAS-based systems. In the DRMS implementations, Foster says that encrypted content "is packaged...and **loaded** onto media server 14," from which the encrypted content can be "access[ed]" by a "customer." Foster, [0049], FIG. 1. That is, after Foster's processor performs Foster's encryption technique to package the encrypted content, the processor executes code to load (i.e., **store**) the encrypted content on the media server's storage device so that, at some later time, users can request that the media server's storage device store the encrypted content (via streaming and/or direct downloading) in another storage device. That too meets limitations [1A] and [1F] under SFI's view of the claim scope that requires the processor to instruct a storage device to store a *data chunk*.

69. For example, Foster's "Content Owner" includes a "movie studio" (Foster, [0069]) that produced content that is stored, rather than consumed as

requires the processor to instruct a storage device to store and/or whether claim 1 requires the processor not to be "agnostic" about whether content is stored by a storage device).

“live” content – movies are not viewed “live,” they are recorded (usually over an extended period time lasting days or even months). Thus, at least in Foster’s DRMS embodiments, Foster’s processor is executing code “to perform storage” at storage devices in servers coupled to the processor. And thus Foster’s processor is **not** agnostic about whether or where content is stored. Instead, as part of Foster’s DRMS content-delivery system, the processor’s code instructs the media server to store the encrypted content in a storage device coupled to Foster’s processor.

Foster, [0049], FIG. 1 (which I have annotated below to show where Foster’s teaches storing encrypted content in the memory of storage devices coupled to Foster’s processor); DI, 25 (“Foster teaches a PC or set-top box storing a version of the data set”).



70. Thus, after encrypting content, Foster's processor executes code that instructs the media server to load (i.e., **store**) the encrypted content on the media server's storage device (which is coupled to the processor). Those instructions are executed by Foster's processor for the purpose of storing encrypted content (1) at the server's storage device (coupled to Foster's processor) and subsequently (2) at end-user storage devices (also coupled to Foster's processor) in response to requests to have the media server's storage device download content to another storage device (e.g., FIG. 1's content "stored" at media server 14 and at PC 18).

71. My opinion in this regard is corroborated, for example, by EX1098, which discloses that with DRM protection (which Foster explicitly provides), even when movies are "stored locally on the consumer's viewing system, the copyright owner has not lost control of the content because each movie...**is stored in a form that the copyright owner dictates.**" EX1098, 5:44-65. Rubin-Depo., 83-89. Similarly, when Foster's encryption method is used to prepare the content for use in a DRMS system (Foster, [0005]-[0006], [0015], [0049]-[0052], FIG. 1), Foster's processor is executing code for storage in a form dictated by Foster's processor. Thus, that is another example of how Foster's processor is not agnostic to whether content is stored: the code being executed by Foster's processor is executing instruction to process the content to be stored in a way that the content owner (and/or content provider) directs.

72. I note too that at his deposition, Dr. Rubin acknowledged that POSAs knew the prior art included digital right management systems where the “purpose of digital rights management hasn’t changed” from the prior art until today, and that CAS systems were used for more than live events, “basically any play that you have permissions and you want to prevent people who aren’t authorized from getting to something,” like video-on-demand services. Rubin-Depo., 83-86. Dr. Rubin further acknowledged at deposition that the prior art included DRM system that “encrypt[ed]” content “before it’s transmitted to the consumer.” Rubin-Depo., 96-99. As Dr. Rubin testified, it was known that DRM was used to “protect[] content on any device with various distribution models, offline, online, with or without return channel.” Rubin-Depo., 96-97. POSAs would have understood that Foster’s processor is using Foster’s encryption methods to prepare content for both offline and/or online viewing.

73. ***DRM for offline viewing:*** For example, Foster’s recognizes that the “program stream is generally used in more reliable transmissions such as those with error correcting codes or **in fixed storage.**” Foster, [0073]. Thus, for example, the processor would execute code to encrypt and encode content for a program stream stored in a DVD disk, which is a form of optical storage using an optical storage device. *See, e.g.*, EX1001, 66:39-44 (identifying an optical disc as a type of storage device). Dr. Rubin himself stated in his declaration for the ’854

patent (EX2033, ¶140 in that matter) that “Foster’s system focuses upon streamed content (**or content stored on a DVD**).”

74. I agree that one implementation of Foster would be storing encrypted content to a DVD (the memory of a DVD storage device). That is, the processor in Foster’s computer is executing code to write/store (so-called *burn*) the encrypted content to a DVD, using a DVD optical storage device *coupled to* Foster’s processor, where the storage device is acting under instructions provided by Foster’s processor.

75. **DRM for online viewing:** As I discussed above and further below, Foster includes embodiments where the processors loads (i.e., stores) encrypted content to a web server, from which users can download content. Foster, [0002]-[0005], [0049], [0073]. And as part of downloading that content, a recipient can stream the content to view as it’s being downloaded (**online viewing** from web-server storage) and stored at the recipient (where it can be viewed **offline** from storage). (Or user can only download and store without viewing the content as it’s being downloaded, another example of storing for **offline** viewing from storage.)

76. As another example of how Foster’s meets limitations [1A] and [1F] even under SFI’s claim scope, I refer to Foster’s disclosure of processing content for a program stream. Foster, [0073]. Consistent with my opinions above, in the example embodiments Foster discloses, Foster’s processor executes code to

prepare encrypted content using one of “two different system options”: “transport stream” (FIG. 14) and “program stream” (FIG. 15). Foster, [0073]. POSAs knew these were the two types of processing options described in the MPEG-2 standard for coding digital media data. Foster explicitly says (as POSAs knew) the program stream “is generally used in more reliable transmissions such as those...**in fixed storage.**” Foster, [0073]. As I noted above, Dr. Rubin has opined that Foster includes storing content on a DVD (just one example of fixed storage). *See* Foster, [0075] (noting that memory 402 may include “optical media”; a DVD is a form of optical media).

77. At deposition, Dr. Rubin acknowledge that the program stream and transport stream are “different ways of encoding MPEG data,” but said he did not “discuss[] the distinctions” in his declarations. Rubin-Depo., 82-83. Though Dr. Rubin did not discuss these distinctions in his declarations, as I noted above and further below, these are distinctions that Foster itself recognizes and are distinctions recognized by exhibits that Dr. Rubin cited and discussed in his declarations.

78. SFI’s EX2014 (at 1) confirms that “[p]rogram streams are designed for...**local file storage.**” *See also* EX2011, [0003] (“**PS [program stream] is preferred to be used in a storage medium....PS is usually stored on discs such as**

DVD[s].”), [0005] (background to “transmit[] a payload of a PES packet through direct memory access” (i.e., from storage devices)).

79. POSAs thus knew that for (at least) Foster’s processing of content for a program-stream, Foster’s processor executes code to instructs a fixed storage device to store the encrypted program-stream data. This is another example of where Foster’s disclosure teaches that the code Foster’s processor executes is not “agnostic” to whether content is for storing. Rather, by executing code to produce an encrypted program-stream (rather than a transport-stream), the processor’s code is not “agnostic” to whether the content is for storage. That is because the program stream is designed for storage.

80. And, consistent with the opinion in my First Declaration (¶¶200-213) and discussed above, Foster’s processor is also executing, and had reason to execute, additional code that instructs a storage device (e.g., a local storage device, a storage device at Foster’s “media server,” and/or a DVD-drive) to store Foster-encrypted data in that storage device *coupled to* the processor. Foster, [0005].

2. It is still my opinion that Foster Renders Obvious Executing Code to Store at least one Data Chunk on a Memory Coupled to Foster’s Processor

81. My First Declaration also demonstrated that POSAs would have been motivated to implement Foster’s processor such that it executes code to store encrypted content in “conventional storage devices coupled to [the] processor,”

e.g., for the benefits of “allow[ing] Foster’s system to store copies of encrypted data for” “backup” and VOD “distribution,” etc., “without having to repeat encryption processing” multiple times for the same data. First Declaration, ¶¶207-212. SFI does not dispute that this implementation of Foster would meet [1F].

82. SFI argues “Petitioner presents no argument as to how (or why) a POSITA would be motivated to modify” Foster’s processor “to execute code to store encrypted content units on” storage devices coupled to the processor. POR, 47. POSAs’ motivation for that obvious implementation of Foster was provided in my First Declaration (¶¶200-213) is further addressed below in this section and in §II.D.2.

83. As to how the code in Foster’s processor would work in that obvious implementation of Foster that I identified in my First Declaration, I note that POSAs understand that SFI’s patent relies on conventional storage devices and SFI does not identify and I do not see in SFI’s patent the type of detail regarding the code implemented by the patent’s processor that SFI now argues need to be disclosed in Foster. For example, the ’609 patent is clear that there is almost no limit on the type of storage devices that it thinks is within the scope of the patent:

The user enrollment data may be parsed and split whereby one or more portions **are stored on** a smart card, a government Common Access Card, **any suitable physical storage device** (e.g., magnetic or optical disk, USB key drive, etc.), or **any other suitable device**.

EX1001, 66:67-67:4; *see also* EX1001, 66:39-44. The '609 patent nowhere discusses code implemented by its processor in order to store content on **any** storage device, let alone “**any** suitable physical storage device...or **any other** suitable device” mentioned by the '609 patent. The lack of detail in the '609 patent regarding “code” corroborates my opinion that POSAs understand the patent to be relying on well-known, conventional techniques of executing code for storing data on any conventional storage device, and thus POSAs would have known how to implement the feature in Foster. Indeed, as shown in Foster’s Figure 8 and discussed in, Foster **already** explicitly discloses that its processor is **already** executing code to store various data in storage devices coupled to the processor, such memory 402, a database 426 (which “may include one or more storage devices”), and external device 408, which POSAs understood could include external storage device. Foster, [0074]-[0078].

84. Thus, Foster’s processor is **already** executing code to store data on a memory of various storage devices that are unquestionably coupled to Foster’s processor, based on instructions provided by Foster’s processor to the storage devices to store various data processed by the processor.

85. Thus, even if SFI were correct that Foster’s processor is not also already executing code to store its *data chunks* on storage devices (which I disagree is not disclosed in Foster), POSAs would have reasonably expected

success in adapting already existing code in Foster’s system that is already being used to store data in a storage device. Moreover, my opinion on the obvious implementations of Foster relied on conventional storage devices coupled to Foster’s processor (e.g., over a bus or network), just as the ’609 patent relies on conventional storage devices and executing conventional code to store *data chunks*. First Declaration, ¶¶173-175, n.12; EX1001, 66:49-55, FIG. 26 (cited First Declaration, ¶124).

86. Furthermore, POSAs would have understood that Limitation [1F] is explicitly recited as software-based (code-implemented) functionality and thus POSAs understood that once the function to be performed has been identified (in this case storing a *data chunk* on the memory of storage device coupled to the processor), POSAs would have reasonable expected success in, and would have known how to, write code to achieve that function using ordinary effort.

87. With respect to [1F]’s limitations regarding the execution of “*code*,” SFI also relies on its arguments regarding [1F]’s “*coupled to*” language (POR, 47), which fail as I discuss below (§II.D).

D. Response to Argument that that Foster Does not Disclose or Render Obvious “Coupl[ing] to the Processor” Storage-Device Memory ([1F])

1. It is still my opinion that Foster Discloses “Coupl[ing] to the Processor” Storage-Device Memory

88. SFI argues Foster’s processor is not “*coupled to*” memory at Foster’s recipients because “there is no indication that [Foster’s] CPU tells the receiver what to do with th[e received] content,” instead “the recipient’s system alone [] decides whether to store the information.” POR, 50-51. In my opinion, SFI’s argument fails because POSAs would disagree.

89. As I explained in my First Declaration, and Dr. Rubin does not dispute, in the context of the ’609 patent, POSAs would have understood that “coupled to” broadly to include a connection using any communication interface that allows two components to exchange information. First Declaration, ¶124 (citing EX1001, 66:49-55, FIG. 26).

90. I note that at deposition, Dr. Rubin similarly testified that “*coupled to*” just means “it’s connected to it.” Rubin-Depo., 190-191. Dr. Rubin further testified that he “ha[s] not come up with” “an opinion” on what “coupled to the processor require[s].” Rubin-Depo., 199.

91. In its POR, SFI identifies no intrinsic evidence requiring the processor to control the claimed *memory*. In its POR, SFI also identifies no intrinsic evidence that says the processor is only *coupled to* the memory of storage devices

it controls/instructs to perform to storage. Furthermore, I also do not see where SFI addresses the evidence I cited in First Declaration that demonstrates that the patent uses “coupled to” broadly to include a connection using any communication interface that allows two components to exchange information. First Declaration, ¶¶124, 204 (citing EX1001, 66:49-55).

92. SFI also admits “Foster’s CPU [*processor*]...encrypt[s] information...for distribution to receivers,” by “sending content units from Foster’s system to the recipient[s],” which “store the transmitted information.” POR, 50-51. That received information is stored in memory (of storage devices) via its connection to Foster’s processor. Rubin-Depo., 192-194.

93. Thus, Foster’s processor is *coupled to* (at least) that memory. First Declaration, ¶¶204-206.

94. As I discussed above (§II.C.1), Foster’s processor is also executing code to store encrypted content in storage devices within Foster’s system, e.g., in Foster’s web/media server, in DVD optical storage devices, **and** at recipient storage devices. I further note that even as to a recipient storage device, Foster discloses that those recipients could storage devices of end users (e.g., storage devices in set-top boxes and/or computers) but could **also** be storage devices under the control of the same or another content service provider. For example, Foster at [0041] defines a “Recipient” as “**any entity, such as a content service provider**

or a receiver, capable of receiving transmissions”). And Foster at [0048] discloses that content processed by Foster’s processor is “**stored by[] a recipient.**” Putting those together (along with the rest of Foster’s disclosure that I discussed in my First Declaration and in this Reply Declaration), Foster is teaching that one of Foster’s embodiments that use Foster’s encryption method, includes Foster’s processor (at a content service provider) executing code for encrypting content and storing the encrypted content in a storage device of a content service provider. That is, the same content service provider is using/controls Foster’s processor to execute code to encrypt content and store it at the content service provider’s storage devices coupled to the processor (like the content service provider’s media server or to a DVD optical storage device).

95. All of the storage devices in Foster’s system are *coupled to* Foster’s processor via a bus and/or interface connection (like a network interface). Foster, [0049], [0074]-[0078], FIGs. 1-2, 16. First Declaration, ¶¶204-213.

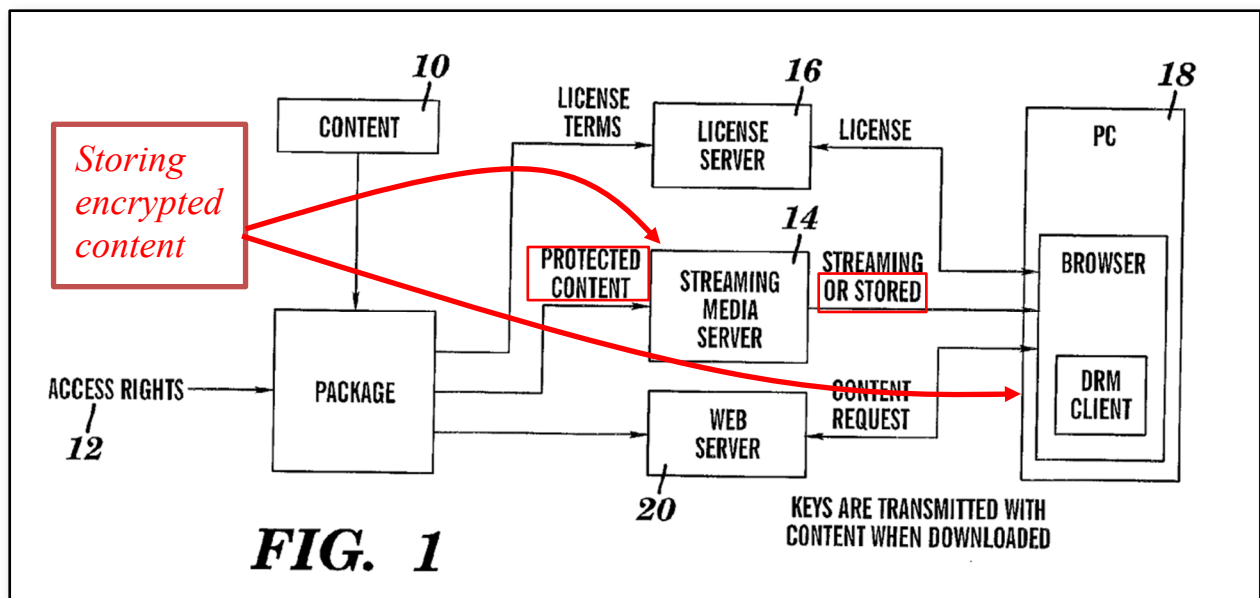
96. I note that Dr. Rubin testified at deposition that the encrypted “content gets from Foster’s processor to the memory and [sic: in] the receiver.” Rubin-Depo., 190-191. Thus, that memory is *coupled to* Foster’s processor under the claim scope because there is a connection using a communication interface that allows those two components to exchange information. First Declaration, ¶124.

97. SFI says the Petition argued Foster’s coupling is only via “bus 410.” POR, 51. My opinions regarding how Foster’s processor is coupled to the memory required by the claims was not based on only bus 410. Instead, I opined that Foster’s communication link between the processor and memory external to the computer 400 is “via ‘**transmission system 424**’ and ‘bus 410.’” First Declaration, ¶¶205-206, n.13 (citing Foster, [0040], [0052], [0075], [0078]-[0079], FIG. 16).

98. SFI’s POR ignores the identification of transmission system 424, which provides the communication link from Foster’s computer to recipients/storage devices; bus 410 is just a conventional component for moving data **within** Foster’s computer, e.g., moving data from the processor to the transmission system or to a local storage device. Bus 410 is only *part* of the connection (and thus the coupling) between Foster’s processor and recipient memory that I identified.

99. For example, below again is an annotated version of Foster’s FIG. 1, which I have annotated in response to SFI’s and Dr. Rubin’s arguments disputing that Foster’s processor is *coupled to* the memory of storage devices that are storing Foster-encrypted content. As I have illustrated below, Foster’s teaches storing encrypted content in the memory of various storage devices coupled to Foster’s processor, such as the “streaming media server 14” and PC 18. The Foster-

encrypted content that is stored in the streaming media server is either stored at the direction of Foster's processor and/or is retrieved from a storage device where the processor directed the storage of content packaged using Foster's encryption technique (i.e., the arrow below that runs from the "PACKAGE" to the "STREAMING MEDIA SERVER" supports the processor storing the packaged content directly to the server and/or the server being provided content that is in another storage device where the processor stored the encrypted content).



100. Then, once the encrypted content has been loaded to (i.e., stored on) the media server, the content can be stored in a stored device at PC 18. That PC 18 can be an end-user's computer (e.g., a subscribed to a VOD service) but it can also be "any entity[']s" PC, including for a content service provider, as I discussed above. Foster, [0041]. For example, the CSP's processor is encrypting content for

(a) that CSP's storage devices (e.g., for backup and/or future delivery to end-users) and/or (b) another CSP's storage device (e.g., another VOD-service provider).

101. All of these implementations of Foster's system include storage devices that are coupled to Foster's processor to receive and store content processed by Foster's processor, and they are all consistent with what Foster expressly disclosed and my First Declaration's opinions demonstrating that Foster anticipates claim 1 (*see* First Declaration, ¶¶200-213).

**2. It is still my opinion that Foster Renders Obvious
“Coupl[ing] to the Processor” Storage-Device Memory**

102. My First Declaration also demonstrated that POSAs had reason to store Foster's encrypted content in “storage devices coupled to Foster's processor” for the benefits of “backup,” “video-on-demand” (VOD), and/or “distribution to later-identified recipients.” First Declaration, ¶¶206-213 (citing corroboration). SFI does not dispute those were known reasons to store encrypted content, but argues they would not “solve any problem[s]” in Foster or for content owners/providers because the Petition-identified “benefits are” allegedly already “provided by the content owner.” POR, 52-54. In my opinion, SFI arguments fail to account for the opinions and evidence I discussed in my First Declaration and POSAs understanding of Foster and the obvious implementation thereof that I identified.

103. As an initial matter, I note that Dr. Rubin admitted at deposition that “a hard disk” attached to a computer is “coupled to” its processor. Rubin-Depo., 195-196. A hard disk attached to Foster’s processor is an example of a conventional storage device that I identified as part of the obvious implementation of Foster that I discussed in my First Declaration (§§206-207).

104. As to SFI’s arguments regarding an alleged absence of need for storing in Foster, in my opinion, POSAs would not have needed to identify a problem in Foster’s system to have had reason to implement the obvious modifications to Foster’s system that I identified in my First Declaration, and to do so with a reasonable expectation of success.

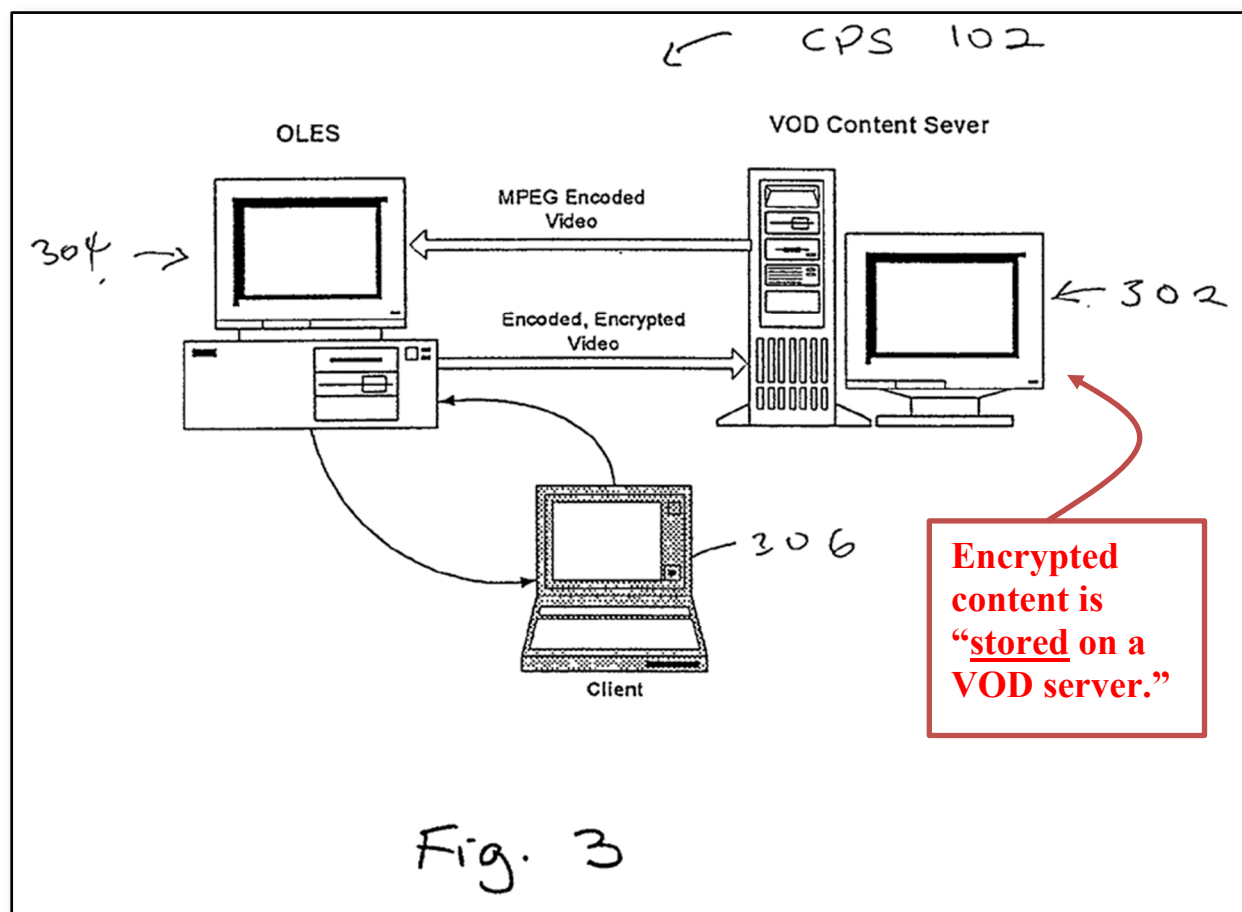
105. SFI argues that content owner/providers sometimes store content themselves, but even if that were sometimes true, it would not have in any way discouraged or negated the benefits of content service providers (“CSPs”) storing encrypted content too, for the reasons I discussed in my First Declaration. For example, POSAs knew that CSPs (e.g., cable-TV) companies often served content they received from content owners/providers (e.g., movie studios). Moreover, CSPs typically encrypted and stored that content before sending to customers, rather than repeatedly encrypting on-the-fly for each VOD request, for various benefits like those I identified in my First Declaration (§§208-213; *see also id.* §§190-193). For example, my opinion in that regard is corroborated by EX1100,

which notes as background information that prior art services, like “video on demand (VOD)” had “need for secure delivery of content,” where “a content provider must encrypt the content” first. EX1100, [0012]-[0013]. EX1100 further notes as background information that “encrypt[ing] content in real time as it is delivered to the customer” was “not always desirable or feasible” and so “there is a need...for content provider to be able to encrypt content before it is transmitted over the Internet as opposed to encrypting it in real time,” which “would reduce cost and overhead that are associated with real time encryption.” EX1100, [0013].

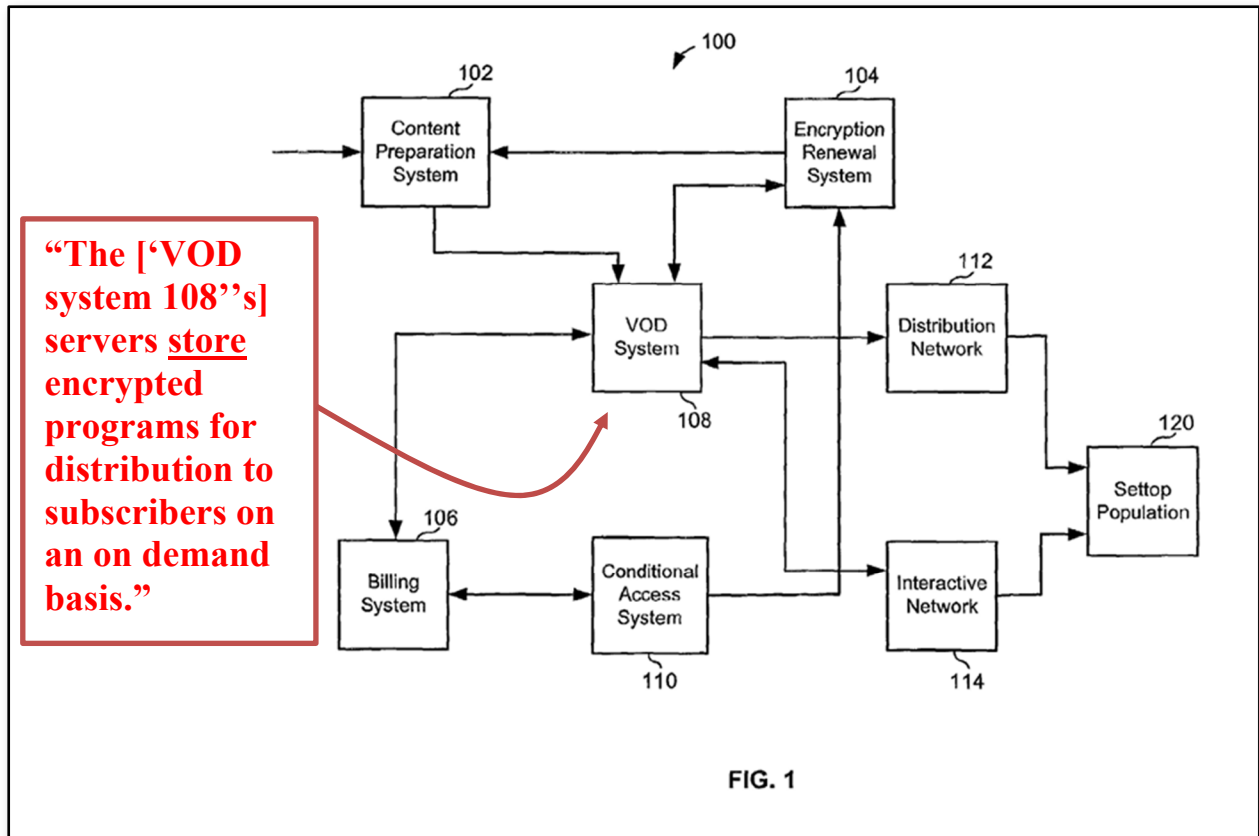
106. My opinion that CSPs in the prior art conventionally stored encrypted content is further corroborated by prior art I cited in my First Declaration (EX1059, EX1060) and the additional prior art I discuss below (and discussed above).

107. In response to SFI’s argument and Dr. Rubin’s opinion that POSAs would not have been motivated to have Foster’s processor execute code for storing Foster-encrypted content in the memory of a storage device coupled to Foster’s processor, I also cite EX1073. EX1073 at [0001]-[0004] corroborates by opinion that POSAs knew that “conditional access” systems were used for VOD system that stored content at storage devices under the control of a content service provider (e.g., an entity that Foster provides as an example entity possessing Foster’s encryption system, including Foster’s processor). EX1073 at [0043]

further corroborates that it was known that VOD systems had “pre-encrypt[ed]” content and “**stores** the content for forwarding...when authorized” for delivery to a storage device. *See also* EX1073, [0098] (“The encoded file is encrypted and verified prior to writing the encrypted material to VOD content server 302..... A streaming mode of pre-encryption is also possible in which content is “streamed” from a source of raw content such as a video (possibly analog tape) through an MPEG encoder, sent to the OLES to *perform encryption and finally stored on a VOD server.*”), FIG. 3 (annotated below).



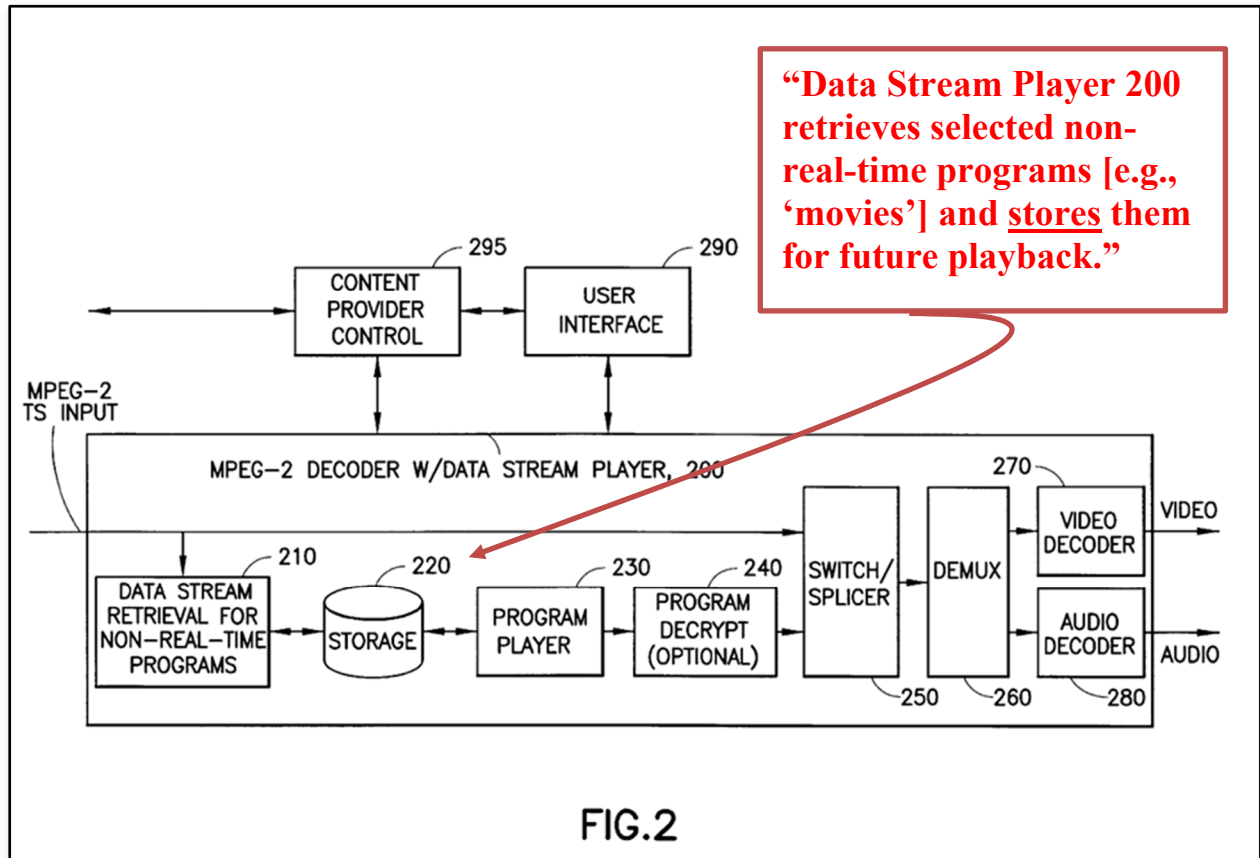
108. In response to SFI's argument and Dr. Rubin's opinion that POSAs would not have been motivated to have Foster's processor execute code for storing Foster-encrypted content in the memory of a storage device coupled to Foster's processor, I also cite EX1097. EX1097 at 2:31-53 describes as background knowledge that using "real-time encryption" in VOD system is problematic and "poses much greater cost and space issues." Thus, rather than use real-time encryption, it was known to implement content-delivery systems (like Foster and Hayhurst) that "that **encrypts content offline (typically before the content is requested by the user)** before it is distributed to point-to-point systems such as cable systems. The system allows content to be encrypted once, **at a centralized facility**, and to be useable at different point-to-point systems." EX1097, 2:31-53, FIG. 1 (annotated below). That is, the content is encrypted and **stored** at the "centralized facility" so it could be delivered (again and again if popular content). Thus, EX1097 is also consistent with my opinion in my First Declaration that POSAs would have wanted to implement Foster to store encrypted content in the memory of a storage device coupled to Foster's processor. First Declaration, ¶¶206-213.



109. In response to SFI's argument and Dr. Rubin's opinion that POSAs would not have been motivated to have Foster's processor execute code for storing Foster-encrypted content in the memory of a storage device coupled to Foster's processor, I also cite EX1075. EX1075 at 1:51-67 teaches that it was known to implement the "MPEG-2 transport" stream in a "system [that] should allow the programs and other content to be provided *on a non-real-time basis for later use by a recipient*. Such content should be able to be easily received, *recorded on a digital VCR or otherwise stored*, and played back at the user's convenience."

EX1075 at 1:51-67 further confirms that it was desirable to have "the delivery of *pre-encrypted content ... in non-real-time*." See also EX1075, FIG. 2 (annotated

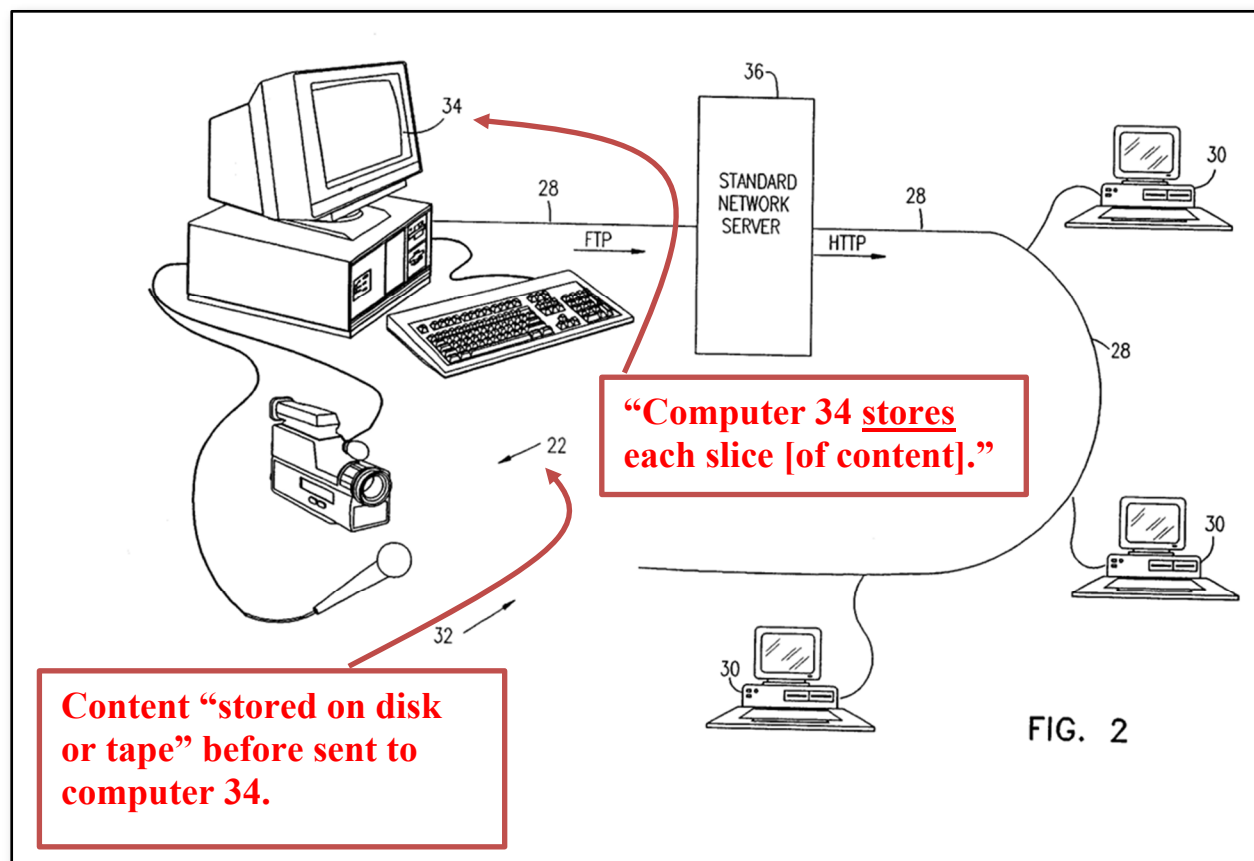
below). Thus, EX1075 further corroborates my opinion that POSAs not only understood Foster’s program stream to be beneficially and primarily directed to storage, but that it was beneficial to *non-real-time* storage and delivery of encrypted content (i.e., *not* live streaming).



110. In response to SFI’s argument and Dr. Rubin’s opinion that POSAs would not have been motivated to have Foster’s processor execute code for storing Foster-encrypted content in the memory of a storage device coupled to Foster’s processor, I also cite EX1076. EX1076 at 9:53-10:5 discloses a prior art system where even “Broadcast data are [] input to the computer, for example, from input

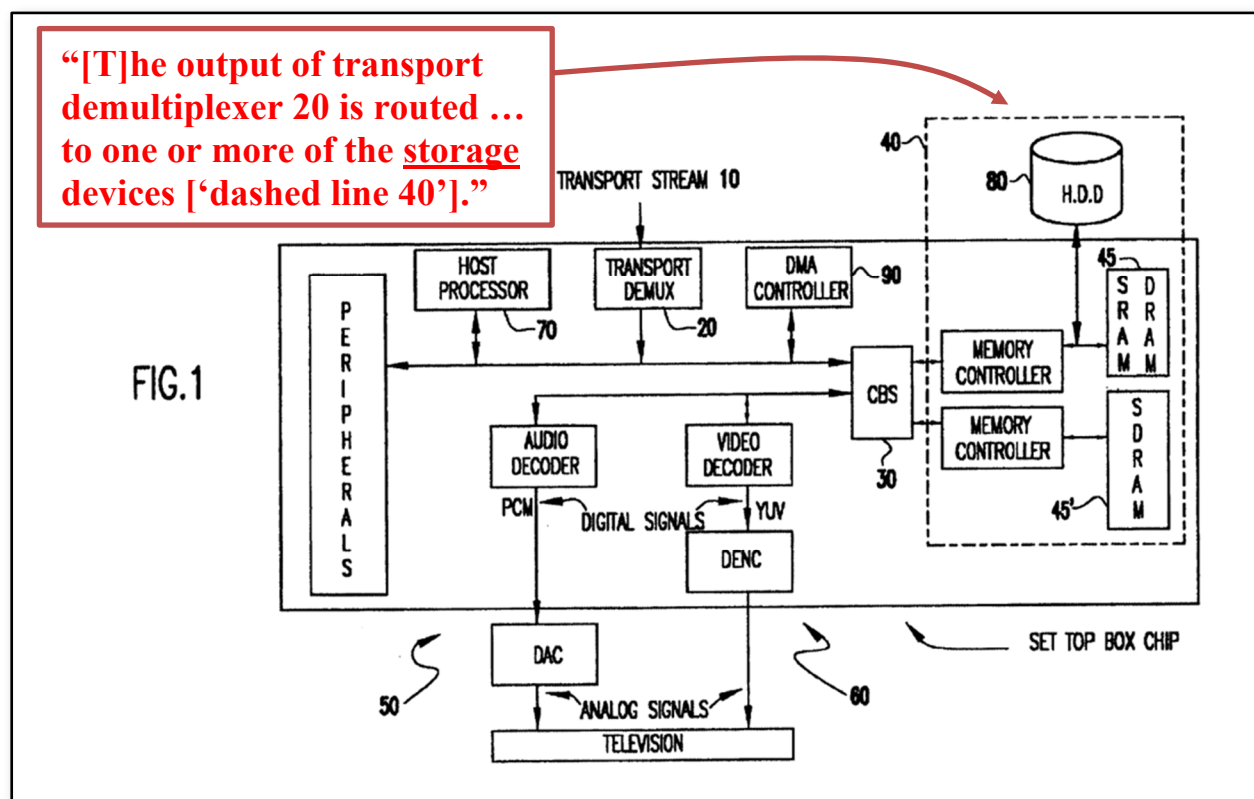
devices 22, or from a video, audio or animation sequence *stored on disk or tape.*”

That is, even “live” broadcasting of content is rarely real-time, but instead often retrieved from a *stored device* that stores the content for some later use.



111. In response to SFI’s argument and Dr. Rubin’s opinion that POSAs would not have been motivated to have Foster’s processor execute code for storing Foster-encrypted content in the memory of a storage device coupled to Foster’s processor, I also cite EX1077. EX1077 (which is another patent by several of the Foster inventors), which discloses that it was known in the prior art to have “pay-per-view movies” that were viewed “at a time convenient” to the user. EX1077,

1:12-21. In those system, users typically had a “set-top box (STB)” (as in Foster, [0040], [0052]), that was expected to have “substantial storage within the STB” and/or paired with “further storage on an onboard mass storage device such as a hard disk drive or compact disk recorder/playback device.” EX1077, 1:48-65. EX1077 at 3:56-5:8 goes on to describe processing of an MPEG “packetized elementary stream” (as in Foster) for “stor[ing] to mass storage.”



112. Thus, it is clear from Foster’s disclosure (and reinforced POSAs’ background knowledge) that Foster’s technique was designed for use in CAS-based systems that included the storage of content both before and after delivery of

the encrypted content to end-user recipients (both for streaming or non-streaming applications).

113. In response to SFI's argument and Dr. Rubin's opinion that POSAs would not have been motivated to have Foster's processor execute code for storing Foster-encrypted content in the memory of a storage device coupled to Foster's processor, I also cite EX1105. Like Foster, EX1105 discloses that an example of a content provider is "a movie studio"¹ and an example of service provider is a "cable company." EX1105, [0034]-[0037]. EX1105 recognizes (as POSAs knew) that "[I]n many scenarios, the service provider 24 provides the requested content off of or **from a storage device is local to the service provider 24.**" Foster, [0034]-[0037]. Thus, EX1105 further corroborates my opinion that POSAs had reason for Foster's processor – the processor at the CSPs system – to store the encrypted content on a local storage device (e.g., for purposes of VOD, later and/or repeated delivery to consumers, etc.).

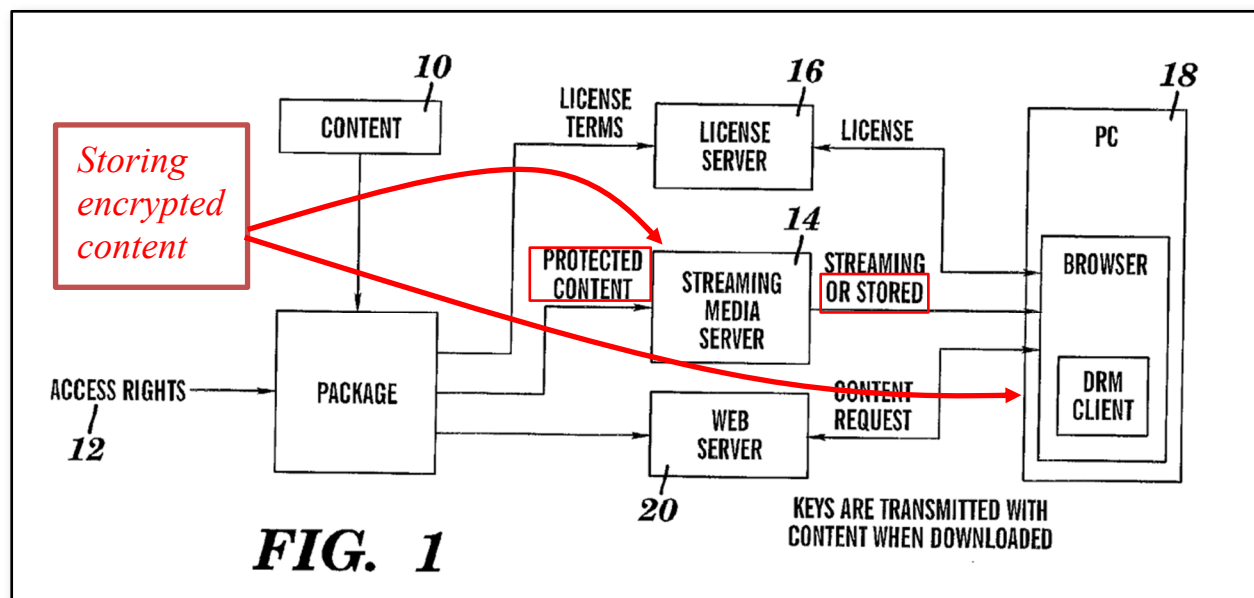
114. In response to SFI's argument and Dr. Rubin's opinion that POSAs would not have been motivated to have Foster's processor execute code for storing Foster-encrypted content in the memory of a storage device coupled to Foster's processor, I also cite EX1080. EX1080 is a journal article from 2003 obtainable from IEEE Xplore, entitled "Pushing the internet direct to the user - security

issues.”⁶ It describes a system where “[d]ata is encrypted at the source by the TEK” and the system “distribute[s] the key to each receiver,” and notes that the TEK “must itself be encrypted” using a KEK. EX1080, 6. EX1080 recognizes (as POSAs knew) “Streamed content...may be playout of pre-recorded content.” EX1080, 3.

115. Indeed, as discussed above (§§II.A-II.B), Foster already discloses encrypting content using an MPEG program stream, which “is generally used...in fixed storage.” Foster, [0073]; EX2014, 1 (“[p]rogram streams are designed for...**local file storage**”); EX1109, 1-6 (“[P]rogramme stream is....intended for the storage and retrieval of programme material from digital storage media.”); EX1001, 66:36-67:11.

⁶ IEEE Xplore’s website lists EX1080 as having been added to IEEE Xplore on “24 November 2003” and published in “IEE Seminar on IP Over Satellite - The next Generation: MPLS, VPN and DRM Delivered Services, 2003.” *See* <https://ieeexplore.ieee.org/abstract/document/1246598>. Based on the above evidence and my review of the indicia appearing on EX1080 and my understanding of the IEEE and IEEE Xplore, it is my opinion that EX1080 was publicly accessible in the prior art by being disseminated or otherwise made available to POSAs exercising reasonable diligence by 2004.

116. For example, as I discussed above (§II.A-§II.B), Foster’s encrypted content is “loaded onto media server 14” for later “download[ing] to PC 18.” Foster, [0049]; FIG. 1 (annotated below).



117. As I discussed above (§II.C.1, §II.D.1), this embodiment of Foster anticipates claim 1. But even if it were not part of an anticipating embodiment, Foster’s disclosure of loading content to a media server for future downloading and storing to a storage device at a recipient’s computer is at least suggesting and motivating POSAs to implement Foster’s processor to execute code to store Foster-encrypted content in a storage device *coupled to*, and acting under the directions of, Foster’s processor.

118. Again, Foster explicitly says the “content service provider” (cable-TV provider) can be the recipient, rather than the (at least initial) recipient from

Foster's processor be an end-user consumer (cable-TV customer). Foster, [0061]. And so that too motivates and suggests implementing Foster's processor at the CSP to store encrypted content on storage devices coupled to the processor before delivery to customers, which meets [1C].

119. Similarly, as I opined in my First Declaration (¶209), one obvious implementation of Foster would have been to encrypt “**content stored by the content provider,**” for the purposes of VOD (and backup). As another related variant of that obvious implementation of Foster that I discussed was supported by EX1059, which taught that rather than a CSP offering VOD, it was also known in the prior art to have “on-demand streaming download of content from the content provider [i.e., a content owner].” EX1059, 2:15-24; *see also* First Declaration, ¶209. In other words, a content owner would implement Foster's encryption method (i.e., the content owner would control the processor) so that the encrypted content would be stored on a VOD-server under the content owner's control, rather than a server under a CSP's control. In other words, the content owner is **also** a content service provider (in the same way that Time Warner was both an owner / maker of content and owned cable companies that delivered content to customers). *See, e.g.*, EX1059, 5:4-23 (corroborating evidence that some content owners in the prior art were also CSPs).

120. SFI argues the corroborating evidence does not “teach[] storage of content by an intermediary.” POR, 54. I disagree. Storage of content by intermediaries is taught by several of the prior art references I cited in my First Declaration (and references that SFI cites and additional corroborating references discussed below).

121. In response to SFI’s and Dr. Rubin’s arguments that POSAs did not have reason to have Foster’s processor execute code for the purpose of storing content at a CSP’s storage devices (or another intermediary to which the CSP has engaged to store content for delivery to the CSP’s customers) I cite as corroboration prior art I cited in my First Declaration (*e.g.*, EX1059, EX1060) and the additional corroborating prior art references discussed below.

122. An example I cite in response to SFI’s arguments and Dr. Rubin’s opinions is EX1060 (cited First Declaration, ¶184). EX1060 describes prior-art VOD systems, where “servers...**store** all movies” for “Cable ‘On Demand’” (*i.e.*, store for an intermediary CSP). EX1060, [0007]-[0008], FIG. 6 *see also id.* [0005]-[0016], [0176]-[0187] (noting that “server 600...[is] operated by a service provider” and stores content “transported” from “source provider 608” (*e.g.*, “movie publisher”)).

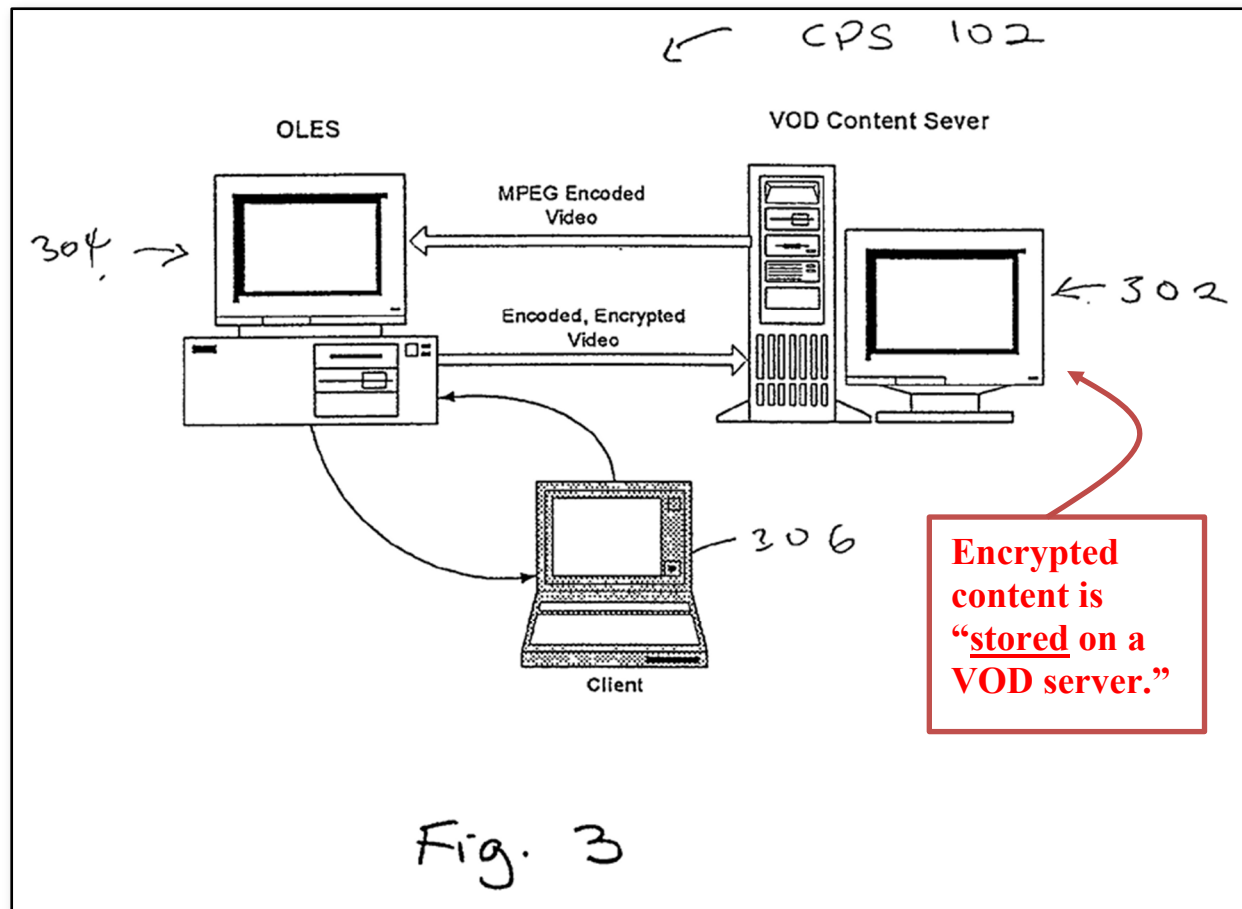
123. Another example I cite in response to SFI’s arguments and Dr. Rubin’s opinions is EX1062 (cited First Declaration, ¶184). EX1062 discloses as

background information that “[v]ideo on demand, such as through cable and satellite providers” include “head end hardware” ([0002]), which is where content is stored from VOD. EX1062 shows “a traditional client/server type of media-on-demand network,” where the central server “may be a regional server or local hub” that “delivers the content to the requesting user.” EX1062, [0005]. EX1062 further notes that the “content [is] available from peers and centralized storage.” EX1062, [0018]. *See also* EX1062, [0043] (noting that “[t]he user may store the content on the device for any period of time the user desires”).

124. As another example I cite in response to SFI’s arguments and Dr. Rubin’s opinions is EX1037 (cited First Declaration, ¶184). EX1037 notes as background information that the prior art includes “Content Delivery Networks,” that POSAs knew where systems having servers that stored content to support the delivery of content-service-provider content. EX1037, [0005]. EX1037 at [0019]-[0025] notes benefit of “transfer[ring] by downloading data from multiple sources or channels in parallel” (as in Hayhurst). One of the (several) benefits of those delivery systems include the support for “Streaming” where “one can actually watch a video as it is being downloaded from multiple sources in parallel,” such as for “Video-on-Demand.” EX1037, [0022].

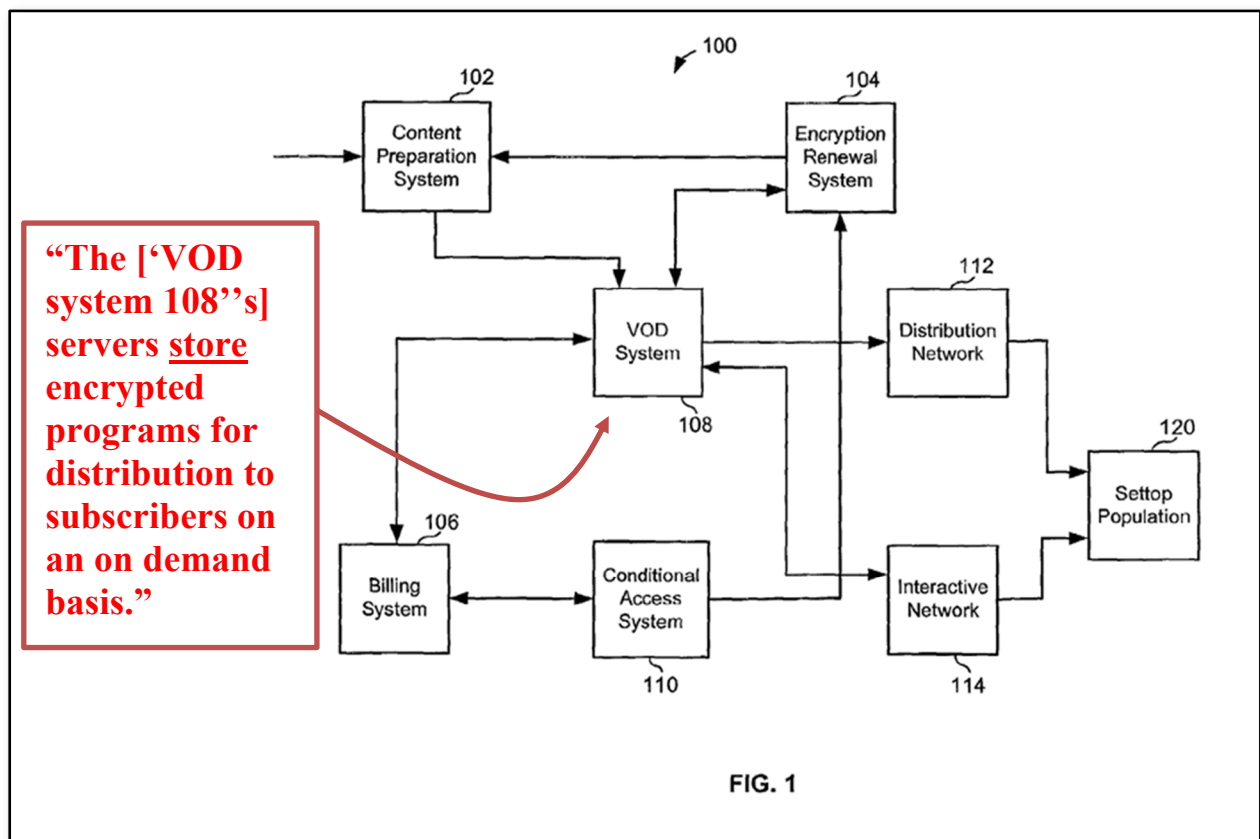
125. As another example I cite in response to SFI’s arguments and Dr. Rubin’s opinions is EX1073. EX1073 at [0001]-[0004] corroborates my opinion

that POSAs knew that “conditional access” systems (like those discussed in Foster) were used for VOD system. EX1073 at [0043] further corroborates that it was known that VOD systems had “pre-encrypt[ed]” content and “**stores** the content for forwarding...when authorized” for delivery to a storage device. *See also* EX1073, [0098] (“The encoded file is encrypted and verified prior to writing the encrypted material to VOD content server 302..... A streaming mode of pre-encryption is also possible in which content is “streamed” from a source of raw content such as a video (possibly analog tape) through an MPEG encoder, sent to the OLES to **perform encryption and finally stored on a VOD server.**”), FIG. 3 (annotated below).



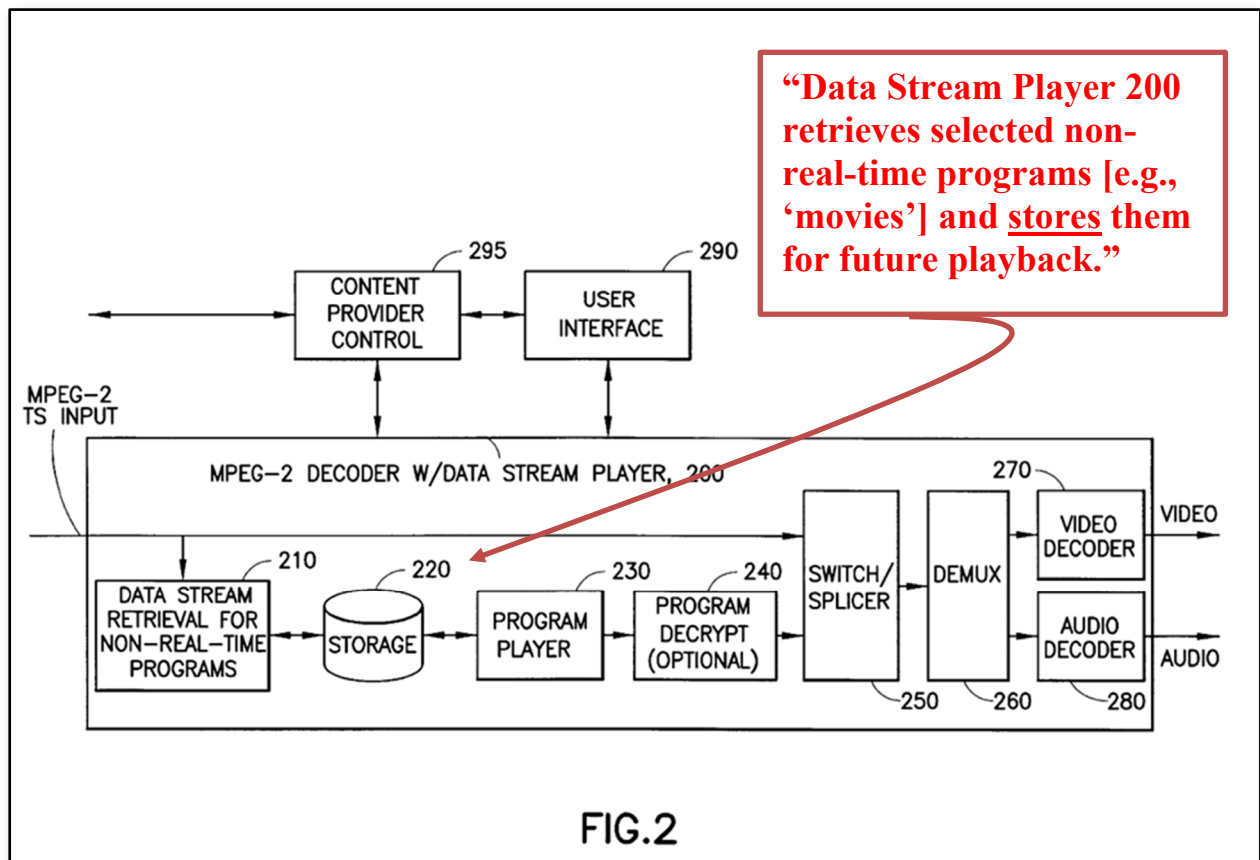
126. As another example I cite in response to SFI's arguments and Dr. Rubin's opinions is EX1097. EX1097 at 2:31-53 describes as background knowledge that using "real-time encryption" in VOD system is problematic and "poses much greater cost and space issues." Thus, rather than use real-time encryption, it was known to implement content-delivery systems (like Foster and Hayhurst) that "that **encrypts content offline (typically before the content is requested by the user)** before it is distributed to point-to-point systems such as cable systems. The system allows content to be encrypted once, **at a centralized facility**, and to be useable at different point-to-point systems." EX1097, 2:31-53,

FIG. 1 (annotated below). That is, the content is encrypted and **stored** at the “centralized facility” so it could be delivered (again and again if popular content). Thus, EX1097 is also consistent with my opinion in my First Declaration that POSAs would have wanted to pursue Foster-Hayhurst for the various benefits that POSAs would have recognized, such as improving the aggregate speed of content delivery and fault tolerance. First Declaration, ¶¶182-185.

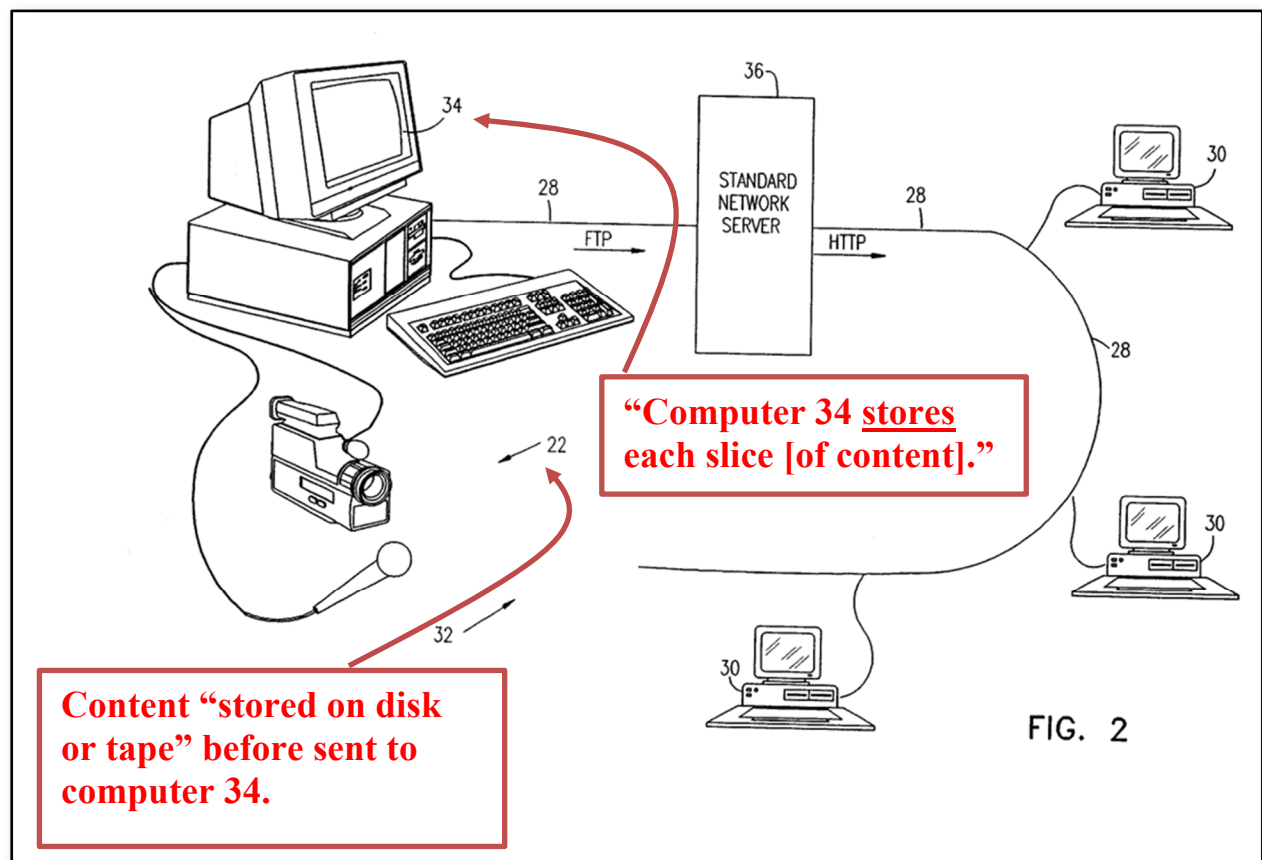


127. As another example I cite in response to SFI’s arguments and Dr. Rubin’s opinions is EX1075. EX1075 at 1:51-67 teaches that it was known to implement the “MPEG-2 transport” stream (discussed in Foster) in a “system [that] should allow the programs and other content to be provided **on a non-real-time**

basis for later use by a recipient. Such content should be able to be easily received, **recorded on a digital VCR or otherwise stored**, and played back at the user's convenience." EX1075 at 1:51-67 further confirms that it was desirable to have "the delivery of **pre-encrypted content ... in non-real-time.**" *See also* EX1075, FIG. 2 (annotated below). Thus, EX1075 further corroborates my opinion that POSAs not only understood Foster's program stream to be beneficially and primarily directed to storage, but that it was beneficial to **non-real-time** storage and delivery of encrypted content (i.e., **not** live streaming).



128. As another example I cite in response to SFI's arguments and Dr. Rubin's opinions, EX1076 at 9:53-10:5 discloses a prior art system where even "[b]roadcast data are [] input to the computer, for example, from input devices 22, or from a video, audio or animation sequence **stored on disk or tape.**" That is, even "live" broadcasting of content is rarely real-time, but instead often retrieved from a **stored device** that stores the content for some later use.

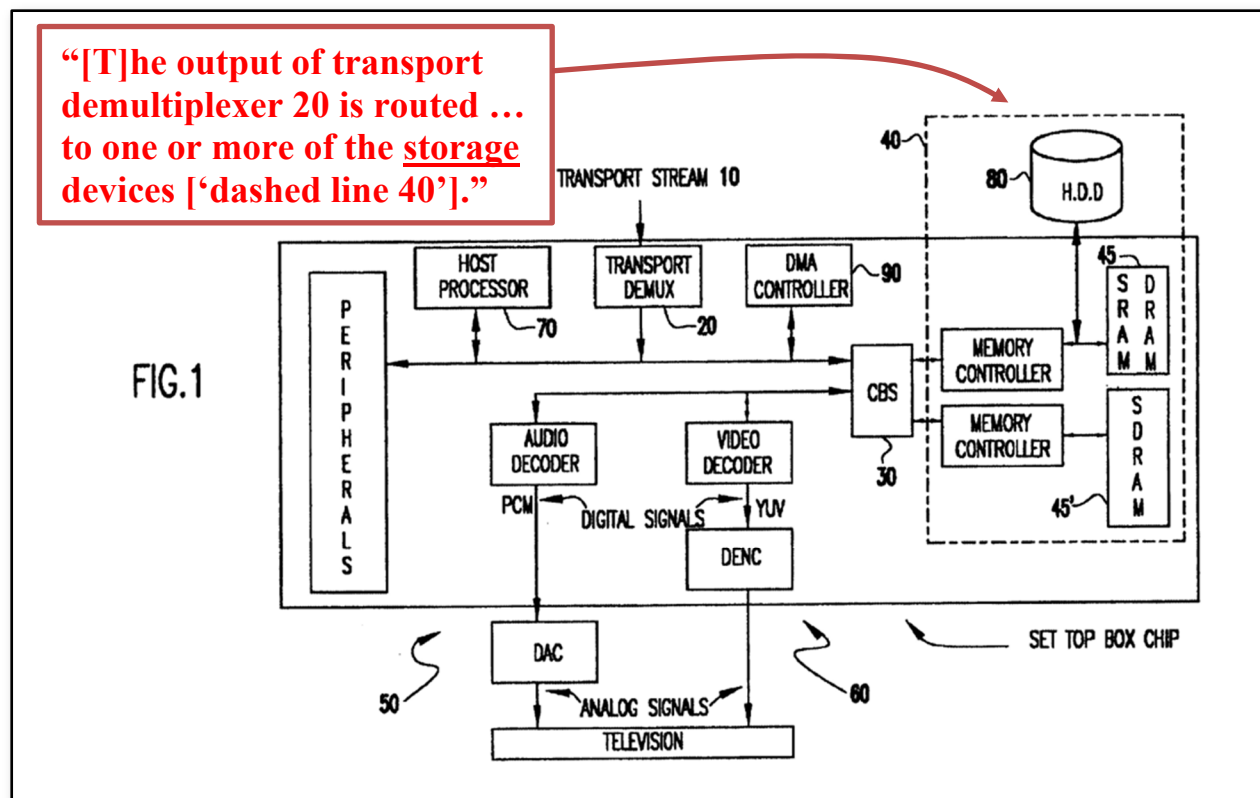


129. As another example I cite in response to SFI's arguments and Dr. Rubin's opinions is EX1077 (which is another patent by several of the Foster inventors). EX1077 discloses that it was known in the prior art to have "pay-per-

view movies” that were stored for viewing “at a time convenient” to the user.

EX1077, 1:12-21. In those systems, users typically had a “set-top box (STB)” (as in Foster, [0040], [0052]), that was expected to have “substantial storage within the STB” and/or paired with “further storage on an onboard mass storage device such as a hard disk drive or compact disk recorder/playback device.” EX1077, 1:48-65.

EX1077 at 3:56-5:8 goes on to describe processing of an MPEG “packetized elementary stream” (as in Foster) for “stor[ing] to mass storage.”



130. Additional examples I cite in response to SFI’s arguments and Dr. Rubin’s opinions are EX1101-EX1102.

- EX1101 discloses as background information that VOD systems were offered by “content aggregator[s] or distributor[s]” using content “obtained” from “one or more content owners.” EX1101 also recognizes that “security” of the content data sets was an important concern for these systems. EX1101, [0004]-[0005]. EX1101 (at [0020]-[0027]) discusses a conventional VOD system having various devices coupled to each other for storing and supplying secured content. For example, EX1101 at [0043]-[0044] describes a “[k]ey wrap” procedure where content owners provide encrypted content to the aggregator that the aggregator can decrypt and secure again on their own before storing at their storage devices at the servers and storage devices in their network.
- EX1102 discloses an “exemplary” system “for protecting a title key in a secure distribution system for recordable media content.” EX1102, [0025]. A “[c]ontent repository” has various “recordable media” (e.g., “software program, music, a movie,” etc) that it stores in various locations and for various purposes. EX1102, [0026]. EX1102 further notes that the content repository “may be part of the same content providing entity” or “content owners may operate [the] content repository.” EX1102, [0026]. A “media recording device” interfaces with the content repository over a communication path (e.g., it communicates with “a personal computer with appropriate software and hardware” or a “special purpose device such as a DVD video recorder”). EX1102, [0027]-[0028]. EX1102 further notes that “cryptographically” protected content is “downloaded from [the] content repository server” – e.g., a user “download[s] content [] from the content repository server 30 to the media recording device 15, which records [i.e., stores] it on physical media 65.” EX1102, [0029]-[0030].

Storing that content allows the user to “play content” when desired. EX1102, [0030]. EX1102 further notes that the content is stored and distributed according to “the constraints of licensing from content owners.” EX1102, [0031]. *See also* FIGS. 1-2, [0032] (describing the “block diagram” of the system wherein a “content repository” has a “content repository server 30 [that] provides content 35 to the media recording device 15 in the form of an encrypted content/title key package 220”).

Thus, it is clear from Foster’s disclosure (and reinforced POSAs’ background knowledge) that Foster’s technique was designed for use in CAS-based systems that included the storage of content both before and after delivery of the encrypted content to end-user recipients (both for streaming and non-streaming applications).

131. I note that at his deposition, Dr. Rubin provided various testimony about the wide-spread use of CAS in the prior art, such as for use in the security of “[f]ile systems” and “video-on-demand” or “basically any place that you have permissions and you want to prevent people who aren’t authorized from getting to something.” Rubin-Depo., 85-86; *see also* Rubin-Depo., 87-89 (testifying that “content producers [] would license the content to the [VOD] hosting sites” and that it was known to have “service providers that would keep the content” that “would be stored,” for example, “[o]n a server”), 90-94 (testifying that VOD system included “intermediaries that would house the actual content” for

distribution to end users). I agree that POSAs would have known that CAS was widely used including VOD / VOD-like systems that are discussed in Foster.

132. Foster's compatibility and suggestion of CAS-based and/or VOD-based system thus corroborates the opinions I expressed in my First Declaration that POSAs had motivation to implement Foster's processor to execute code to store encrypted content in the memory of a storage device coupled to Foster's processor (including based on instructions from Foster's processor to store the content).

133. SFI argues "Petitioner's proposed modifications would" "compromise" Foster's security "or require significant resources to prevent" compromising security because Foster could be "hacked" whereas Foster's content sources/owners "already securely store[]" their content. POR, 55. SFI argues modifying Foster would require "significantly more storage" that "would have to be as secure as the storage at the content provider/owner." POR, 55. In my opinion, POSAs would disagree.

134. SFI and Dr. Rubin offer no evidence that the risk of hacking or the overall storage requirements would have been increased when content is stored by a CSP rather than by a content owner/provider. Nor does SFI allege, let alone support with independent evidence, that CSP had (or were restricted in using) storage devices that were not as secure as (unidentified) storage devices used by

content provider/owner. But even if it added cost or complexity to make CSP devices more secure (and I see no evidence that a conventional CSP would not have had storage devices at least as secure as content owners), POSAs would have still understood that increased cost and/or complexity would not override other considerations that would have motivated using known storage techniques, that would have provided practical benefits to Foster's system (e.g., for CSPs) when modifying Foster in the manner I discussed in my First Declaration.

135. POSAs would also disagree with SFI's argument because POSAs knew that content owners/providers often had intermediaries – CSPs and/or storage providers, like content delivery networks (a type of CSP) – to store their content. Moreover, as I discussed in my Declaration (¶CITE) and above, content owners were known to doubled as an owner of content and a CSP, and thus had need to store their content in encrypted form before delivering the content to a recipient. My opinion in that regard is corroborated by prior art references I cited in my First Declaration, and the additional references discussed below, which include references that SFI placed into the record.

136. In response to SFI's and Dr. Rubin's arguments that POSAs that there is no evidence that content owners in the prior art used intermediaries to store their content, I cite EX1103. EX1103 at [0175] discloses a "Content Server

Node 1500 host[ing – i.e., storing .en] a variety of video content files from different content originators, such as movie studios and sports entertainment.”

137. In response to SFI’s and Dr. Rubin’s arguments that POSAs that there is no evidence that content owners in the prior art used intermediaries to store their content, I also cite EX1104. EX1104 at [0002] discusses prior art “pay per view (PPV) and video-on-demand (VOD) services,” wherein:

Each service offers a viewer a video (e.g., a motion picture) that is delivered for viewing by the viewer who orders and pays for the viewing. The VOD service offers the viewer a video that can be requested at any time and is available for viewing at the discretion of the viewer. Each service can be offered and delivered by wireless communication, such as satellite or cellular telephone services, or by wired communication, such as a cable television service.

138. EX1104 at [0003] notes that “video delivery service provider... receive[] distribution rights from the content provider” so that they can “offer[] video delivery services to viewers via a communication system by the distribution of PPV assets and/or VOD assets.” And EX1104 at [0004] notes that the video distribution services involved numerous servers storing content for delivery to end-users.

139. POSAs would also disagree with SFI’s allegation – unsupported by any independent evidence – that content owners/providers would always securely store content themselves ignores that Foster includes embodiments where content

owners supply **non**-encrypted content to CSPs. Foster, [0079]. Whether the content owner was or was not storing that content before delivering it to CSPs, the CSPs were receiving un-encrypted content. Thus, Foster's encryption of that content at the CSP **added** security to that unencrypted content, rather than reduced the security of the content. In other words, POSAs would have recognized that Foster (let alone the obvious implementation of Foster that I discussed) would **not** "make Foster's system worse." POR, 55-56.

140. But even if there were (hypothetical) downsides to Foster's computer system storing the encrypted content, storing encrypted content on devices directly coupled to Foster's processor would have provided **benefits** to Foster's system that I identified in my First Declaration (and discussed above). POSAs would have understood that it was a commonplace fact in the field of securing data that design decisions entailed making tradeoffs among multiple objectives. Whatever increases in storage space and/or security of storage devices that would have been needed – and I do not believe there would have been as I discussed above – that would have been a matter of design details and/or tradeoffs, from which POSAs would have used their ordinary skill to resolve, with a reasonable expectation of success in doing so.

III. MY REPLY TO SFI REGARDING GROUND 1: IT IS STILL MY OPINION THAT ALL CHALLENGED CLAIMS WOULD HAVE BEEN OBVIOUS OVER ORSINI

141. For Ground 1 (the Orsini-based Ground), SFI only disputes whether Orsini discloses, or an obvious implementation of Orsini would have included, limitation [1E]: “*a cryptographic operation based on the first key to further secure...data chunks.*”

142. My First Declaration advanced several alternative arguments demonstrating how Orsini both anticipates and/or renders obvious claim 1 (including limitation [1E]) based on disclosures in Orsini, POSAs understanding thereof, and/or use of conventional, well-known encryption techniques (all of which I corroborated with multiple prior art references). First Declaration, ¶¶99-123. In my opinion, those opinions are still correct, as SFI’s POR and Dr. Rubin’s declaration did not demonstrated that [1E] (or any other limitations of the challenged claims) was not disclosed and rendered obvious over Orsini.

143. For Ground 1, I have been asked to focus this Reply Declaration on one of my obviousness opinions for Ground 1. Specifically, I have been asked to focus on responding to SFI’s arguments regarding my opinion that POSAs had reason to encrypt Orsini’s share keys with a key-encryption-key (“KEK”). As I discussed in my First Declaration, POSAs had reason to do that to safeguard the share keys from detection, as multiple corroborating references I cited

demonstrated was a well-known, conventional technique in the prior art. First Declaration, ¶¶120-122 (citing corroboration discussed again below, §III.A). SFI does not appear to dispute that encrypting encryption keys with KEKs was known or that encrypting Orsini's share keys with a KEK meets [1E] under SFI's constructions.⁷

144. In my First Declaration (¶120), I also discussed Orsini's disclosures regarding public-key encryption. As I discuss further below (§III.A), the reasoning behind my opinions that it would have been obvious to POSAs to use a conventional KEK to encrypt Orsini's share key was fundamentally different from, and not dependent on, testimony regarding Orsini's disclosure of using public-key encryption. Instead, my obviousness opinions were and are premised on POSAs' background knowledge **and** prior-art references that **explicitly** provided POSAs with motivation to encrypt Orsini's share keys and a reasonable expectation of success in doing so. That obviousness opinion was independent of Orsini's disclosures involving public-key encryption. In other words, even if Orsini said nothing at all about public-key encryption (which Orsini does not use to encrypt

⁷ I have offered alternative opinions as to how Orsini, and Foster, meet and / or render obvious [1E] under a proper claim scope. *See, e.g.*, First Declaration, ¶¶99-123.

share keys), it was and still is my opinion that POSAs would have had reason to, and a reasonable expectation of success to, encrypt Orsini's share keys with a convention key-encryption key (KEK), for its well-known benefits that I demonstrated with corroboration in the prior art. *See also infra* §III.A (responding to SFI's and Dr. Rubin's arguments regarding the obviousness of using a KEK in Orsini).

A. Response to Argument that it Would Not Have Been Obvious to POSAs to Encrypt Orsini's Share Keys Using a Known Key-Encrypting Key ("KEK")

145. In its POR, SFI admits that the '609 copies Orsini's "Figures 1-25 and paragraphs 1-[4]23," which SFI does not disclose most of claim 1. However, SFI alleges that the '609 patent added adds "17 new figures and...~10,500" words of "significantly different technology." POR, 1-2; EX2003, ¶33. SFI does identify which "new" disclosures in the '609 patent are allegedly additional support for the feature claimed in limitation [1E]. I have reviewed the '609 patent, and at most, there only appears to be some cursory, additional disclosure of using a "workgroup key" that encrypts another "encryption key," for the predictable result of (marginal) additional security. EX1001, 77:20-23, 80:24-38, 81:62-65. That is not

“new” or “significant” because as I discussed in my First Declaration (¶¶120-122) and further below (*infra* §§II.A.1-II.A.3), using KEKs were well-known.⁸

146. SFI and Dr. Rubin do not dispute that the encryption method described in reference to Orsini’s Figure 21 results in an encrypted *share* of data being stored with a share-encryption key that is itself *not* encrypted. First Declaration, ¶121. SFI and Dr. Rubin do not dispute that each unencrypted share key in Orsini that is stored with an encrypted share is not the key to unlock that share (e.g., Orsini’s obfuscation step in Figure 21). First Declaration, ¶¶57-58.

147. Notwithstanding that obfuscation step, even SFI recognizes that an attacker nonetheless can “reconstruct [Orsini’s] encrypted dataset if the attacker has access to each of the shares.” POR, 4; EX2033, ¶41. That is because Orsini’s various share keys are not encrypted, but rather have only been non-cryptographically exchanged for another share key. Thus, once an attacker has

⁸ Orsini also discloses implementing protections for workgroups. *See* Orsini, [0293], [0408]-[0413]. To the extent Orsini discloses workgroup-based protection any differently than in the ’609 patent, my First Declaration identified several ways by which there were conventional, known ways to implement workgroup-based protection that would have met [1E] when used in Orsini to perform the workgroup-based protection it sought. First Declaration, ¶¶114-115.

obtained all of the encrypted shares, the attacker has also obtained all of the keys in plaintext form.

148. As I discuss further below (§§III.A.1-3), using the unencrypted share keys to decrypt the data would have been within the skill of a conventional determined attacker using conventional techniques and computing power at the time, even if the attacker was unaware that Orsini’s method used non-cryptographic obfuscation for the share keys. POSAs would have recognized that the same would not have been true if a conventional encryption technique, like a KEK, is used to prevent decryption of encrypted shares. As I discuss further below, POSAs knew that the best practices for security systems were to be able to provide security even when the attacker knows the algorithm for the encryption (because better security comes from keeping the key secret, not the algorithm).

149. In my First Declaration, I demonstrated in detail, with multiple pieces of *corroborating evidence*, that POSAs had reason to encrypt Orsini’s share keys with a KEK to prevent an attacker from reconstructing the encrypted dataset, even when the attacker had access to all of the encrypted shares. First Declaration, ¶¶120-122. The corroborating evidence that I cited was direct in its teaching: “cryptographic keys should *never* occur in clear [*i.e.*, unencrypted] form outside the cryptographic device.” EX1035, 12:15-23 (cited First Declaration, ¶¶120-121).

150. As the corroborating prior art explains, POSAs knew that when an encryption key is transmitted or stored outside of the encrypting device (and often even when stored in a local storage device), then a well-designed encryption system should “encrypt the encryption keys themselves...[with] a key encryption key.” EX1036, 1:66-2:6 (cited First Declaration, ¶121). POSAs understood that the benefits of and desire to encrypt encryption keys with KEKs (so as not to have unencrypted keys outside the device) is true irrespective of whether the “encryption keys are included with the encrypted message” (EX1036, 1:66-2:6) or are stored separately from and/or obfuscated with encrypted data-portions.

151. My opinion regarding POSAs’ background knowledge that it was known (and preferred) to encrypt encryption keys with KEKs, even when the encryption key is not stored with the data it encrypts, is corroborated by several prior art references discussed below.

152. For example, I cited EX1035 (cited First Declaration, ¶¶121-122). EX1035 (at 10:42-62) discloses that the “objectives of key management are: (1) cryptographic keys should **never** occur in clear form outside the cryptographic device, except under secure conditions” and “(2) no cryptographic operation, or combination thereof, using any cryptographical quantities that are routinely stored or routed through the system, or derived therefrom, should permit clear keys to be recoverable outside the cryptographic device.” EX1035 continues that “in keeping

with the first objective, if the system generated terminal keys are to be stored at the host system they **must be protected by being enciphered under another key.**”

EX1035, 10:56-62.

153. As another example, I cited EX1036 (cited First Declaration, ¶122). EX1036 at 1:66-2:6 taught as background information that “[s]ecurity for such a system, where the encryption keys are included with the encrypted message, is not to store or transmit the encryption keys in an easily readable format but rather to encrypt the encryption keys themselves.” That is, “the data encryption key will be encrypted by a key encryption key.” EX1036, 2:2-6.

154. In response to SFI’s arguments and Dr. Rubin’s opinions challenging my opinions on the obviousness of using a KEK in Orsini, I cite as another example EX1086. EX1086 at 7:3-22 discloses a system with an “archive station 4 [that] is usually a server or consists of several servers, i.e. computers providing service,” including at least, “a first and a second server... wherein encrypted data 10 are stored on the first server and **the second keys 7 belonging thereto are stored on a second server.**” That is, the encrypted data and the encryption key for that data are stored in separate locations. EX1086 at 7:3-22 discloses that **“Further the second keys 7 can be stored encrypted on the archive station 4.”** That is, even though the encryption keys are stored separately from the encrypted data, EX1086 corroborates my opinion in my First Declaration that POSAs would

have known how to and would have wanted to encrypt those keys. Encrypting those keys teaches or at least suggests using a KEK for that encryption of keys because KEKs, by definition, are keys that encrypt other encryption keys.

155. In response to SFI's arguments and Dr. Rubin's opinions challenging my opinions on the obviousness of using a KEK in Orsini, I cite as another example EX1087. EX1087 teaches a "method for securely storing data," where the "data are encrypted using a backup key" and "[t]he backup key is protected as a protected backup key" (i.e., the backup encryption key is encrypted with a KEK). EX1087, [0017]. Rather than be stored with the data, "the backup key is protected by being stored" elsewhere, such as "in an internet server, in a pocketable medium or in a trusted computer." EX1087, [0019]. "Preferably, protecting the backup key **includes encrypting the backup key.**" EX1087, [0020]. Thus, EX1087 is another example of prior art demonstrating that POSAs knew that even when encrypted data was stored separately from its encryption key, the encryption key should be protected by a KEK when stored outside the encrypting device (which itself was known to be used in a conventional key management systems). EX1087, [0020]-[0025].

156. In response to SFI's arguments and Dr. Rubin's opinions challenging my opinions on the obviousness of using a KEK in Orsini, I cite as another example EX1112. EX1112 at [0025] discusses the use of a public encryption key,

which it says “is also known as a ‘session key,’” and it is “used to encrypt each individual field of data stored in [a] data store.” EX1112 at [0026] discloses that there is also “master key” that is “protected by a password.” The master key is a key “used” “to generate the public key.” EX1112, [0022]. EX1112 further discloses that “[t]he session key may be stored in the same data store used to store the encrypted data, or in a different data store.” EX1112, [0027]. EX1112 adds that “[w]herever the session key is stored, it is preferable that it is encrypted beforehand.” EX1112, [0027] (e.g., it “may be encrypted using the master key”). Thus, EX1112 corroborates my opinion that POSAs understood that it was conventional to encrypt an encryption key “wherever” it is stored, whether stored with data it is encrypted or stored somewhere else (like a conventional data store).

157. Moreover, Orsini itself says “[i]t is readily apparent to those of ordinary skill in the art that a wide variety of encryption methods or algorithms are suitable for use in the present invention, including, but not limited to several well-known, prior art encryption techniques / standards, such as “Triple DES and AES.” Orsini, [0378]. Those were examples of prior art commercialized and/or standardized encryption techniques. POSAs knew that Triple DES and AES, for example, described using KEKs and the storage / management thereof.

158. My opinion in that regard is corroborated by several prior art references discussed below.

159. In response to SFI’s arguments and Dr. Rubin’s opinions challenging my opinions on the obviousness of using a KEK in Orsini, I cite as another example EX1094. EX1094 at [0056] notes that “[s]tandardized algorithms,” like “Triple-DES **Key Wrap**” were compatible with the referenced “exemplary key management mechanisms.” POSAs understood that “Key Wrap” in this context refers to wrapping (encrypting) an encryption key with another encryption key (i.e., a KEK). EX1094 at [0057] further notes that “[v]arious algorithms can be used to encrypt the content with the encryption key, including the Data Encryption Standard (DES), Triple-DES, RC2 and AES algorithms.”

160. In response to SFI’s arguments and Dr. Rubin’s opinions challenging my opinions on the obviousness of using a KEK in Orsini, I cite as another example EX1090. EX1090 at 19:47-61 discusses “deriv[ing] a key to ‘wrap’ (e.g., encrypt) the data encryption key (DeK),” by “generating a Key Encrypting Key (KeK) 430 to protect the Data Encrypting Key (DeK) 412, that, in turn, is used to encrypt the plaintext data 402.” And at 21:1-22:54, EX1090 discusses using the “AES[] key wrapping function” (i.e., a KEK function) to encrypt the DeKs.

161. In response to SFI’s arguments and Dr. Rubin’s opinions challenging my opinions on the obviousness of using a KEK in Orsini, I cite as another

example EX1096.⁹ EX1096 is a copy of an AES key-wrap algorithm, and it notes that “[t]his key wrap algorithm **needs to provide ample security to protect keys**

⁹ EX1096 is a document published by the IETF – the Internet Engineering Task Force, which is the “standards development organization (SDO) for the Internet.” <https://www.ietf.org/about/>. EX1096 provides on its face a “Copyright Notice” “Copyright (C) The Internet Society (2002)” and lists a date of “September 2002.” In my opinion, POSAs would have been aware of RFC (Requests for Comments) published by the IETF, like EX1096, at the time of the alleged invention of the ’609 patent and would have known how to locate RFCs like EX1096 by navigating to the IETF’s RFC database. As the IETF’s website notes, the “IETF publishes its technical documentation as RFCs,” which “are freely available to download, copy, publish, display and distribute, in a variety of formats, under a license granted by the IETF Trust.” <https://www.ietf.org/process/rfc/>. *See also* <https://bit.ly/3Bw9Rtr> (a February 6, 2007 archived copy of EX1096 from the Internet Archive’s Wayback Machine). Based on the above evidence and my review of the indicia appearing on EX1096 and my understanding of the IETF and its RFC database, it is my opinion that EX1096 was publicly accessible in the prior art by being disseminated or otherwise made available to POSAs exercising reasonable diligence by 2003.

in the context of prudently designed key management architecture.” EX1096, 1-3.

162. Orsini’s use of standardized encryption techniques that already described and accounted for use of KEKs (like AES and DES), reinforces my opinion that it would have been obvious to use a KEK in Orsini **and** that POSAs would have had a reasonable expectation of success doing so in Orsini. First Declaration, ¶¶127-128. *See also infra* §III.B (where I reply to SFI’s arguments regarding POSAs’ expectation of success in modifying Orsini).

163. In contrast to the known cryptographic protection that would have been provided using the known, conventional cryptographic protection of a KEK, Orsini’s non-cryptographic obfuscation only provides (some minor) security when an attacker obtains fewer-than-all shares. In contrast, POSAs would have recognized that a KEK would have provided cryptographic security (generally stronger than non-cryptographic security and certainly stronger than Orsini’s obfuscation) **even if** an attacker obtains all of the encrypted shares.

164. For example, with Orsini’s technique, once an attacker recovered all of the encrypted shares (and thus all of Orsini’s plaintext share keys), it would have been within the capabilities of a conventional attacker (i.e., an attacker having the level of sophistication that POSAs would have wanted to protect against) to reverse Orsini’s obfuscation to put the share keys back together.

165. Indeed, in Orsini’s Figure 21 embodiment, there are only four shares and four share keys, and thus (at most) only 24-different combinations of shares and keys.¹⁰ Moreover, neither Orsini’s Figure 21 method nor the challenged claims require four shares, but instead allow for as few as two shares, which would have only **one way to non-cryptographically obfuscate** in the manner Figure 21 shows, as Dr. Rubin admitted at his deposition. Rubin-Depo., 24-25. *E.g.*, Orsini, [0013] (“The method [of the present invention] comprises one or more steps of parsing, splitting or separating the data to be secured *into two* or more parts or portions.”). Thus, POSAs understood that whatever additional security was provided by Orsini’s obfuscation step, was clearly not significant security, and certainly not comparable to the type of cryptographic security that could be provided by a KEK (whether in place or in addition to obfuscation).

166. In contrast to Orsini’s non-cryptographic obfuscation, with the cryptographic protection of a KEK, even if an attacker obtained all of Orsini’s encrypted shares and keys, the attacker would have **still** needed to know **where** the

¹⁰ Because Orsini’s Figure 21 embodiment with the (optional) ability to obfuscate does not store encrypted shares with their share keys, there are even fewer than 24 combinations (e.g., the six combinations where share 1 is stored with key 1 would not be allowed).

KEK was stored, how to **breach** the security of that key store, **and then determine** how to find the right KEK **and break the cryptographic protection of the KEK**.

167. Using the cryptographic protection of KEK would have also facilitated providing all encrypted shares to trusted parties, while still maintaining stronger control over when and how the parties could access the secured data. In contrast, with Orsini's obfuscation, once a party obtained all of the encrypted shares (which are stored with the keys), POSAs would have recognized that it would have been within the capabilities of a determined attacker to reverse the obfuscation to decrypt the shares. POSAs would have also known that the time it would have taken an attacker to break the security in the shares would have been significantly shorter than the time it would have taken to break KEK-based encryption (which, depending on what encryption was used, may not even have been possible at the time using then-existing encryption techniques for KEKs).

168. My opinion that POSAs' background knowledge would have made it clear to POSAs that Orsini's non-cryptographic obfuscation step provided little, and certainly less, security compared to the cryptographic security of a conventional KEK is corroborated by several references that I cited and discussed in my First Declaration (§§121-122) and the additional ones I cite and discuss below in response to SFI's arguments and Dr. Rubin's opinions.

169. In response to SFI’s arguments and Dr. Rubin’s opinions challenging my opinions on the obviousness of using a KEK in Orsini, I cite SFI’s EX2007. EX2007 teaches that even “26 possible keys” is subject to “a *brute-force attack*” and that’s poor security because “any secure encryption **must not be vulnerable to such brute-force attack.**” EX2007, 11. As I discussed above, Orsini’s example embodiment includes (at most) 24 different combinations (and can be as few as one way to obfuscate, when there are only two shares). POSAs would have recognized that such a small number of combinations would have also been subject to a (relatively simple) brute-force attack when all the shares were obtained, and thus provided substantially weaker security compared to the cryptographic protection of a KEK (whose security would not have been meaningfully reduced, if at all, even if all encrypted shares were obtained).

170. In response to SFI’s arguments and Dr. Rubin’s opinions challenging my opinions on the obviousness of using a KEK in Orsini, I further cite EX1089. EX1089 at [0569] discusses “a non-cryptographic security approach” (e.g., one that “applies a 32-byte XOR mask, repeatedly, to the entire data object”), which is substantially more secure than Orsini’s obfuscation. Nonetheless, EX1089 says that non-cryptographic security does “serve[] to obscure the data so it cannot be used directly, **but is generally more open to attack than is the cryptographic security approach.**” EX1089, [0569]. That is, EX1089 corroborates my opinion

that POSAs would have known that non-cryptographic security (like Orsini's obfuscation, which POSA would have recognized as being weaker than EX1089's 32-byte XOR mask) would also have been not as secure as conventional cryptographic security techniques (like using a KEK).

171. In response to SFI's arguments and Dr. Rubin's opinions challenging my opinions on the obviousness of using a KEK in Orsini, I further cite EX1090. As EX1090 (at 2:39-41) discloses (as POSAs knew), "[t]he more difficult it is for an unauthorized entity to obtain a valid key, the more secure the cryptographic scheme." There can be no reasonable dispute that POSAs would have known that using the cryptographic protection of a KEK could have made Orsini's data more secure (regardless of what security was provided by Orsini's non-cryptographic obfuscation step). Indeed, that is what EX1089 teaches, as I discussed above.

172. In response to SFI's arguments and Dr. Rubin's opinions challenging my opinions on the obviousness of using a KEK in Orsini, I further cite EX1106. EX1106 is a document published by the U.S. Department of Commerce's National Institute of Standards and Technology ("NIST"), titled "Recommendation for Key Management – Part 1: General." It is NIST Special Publication 800-57, published

August, 2005. *See* EX1106, pdf page 2.¹¹ This document provided “basic key management guidance” to advise on the “‘best practices’ associated with key management.” EX1106, 5. As EX1106 states (and POSAs knew), “Keys are analogous to the combination of a safe” and “the security of information protected by cryptography directly depends on the strength of the keys” and “the protection afforded to the keys” themselves. EX1106, 5.

¹¹ NIST’s website indicates that EX1106 was published “August 2005.”

<https://csrc.nist.gov/pubs/sp/800/57/pt1/final>. I have also located a September 11, 2005 archived copy of EX1106 from the Internet Archive’s Wayback Machine.

See <https://bit.ly/3ZQiYwR>; *see also* <https://bit.ly/4fElcFO> (September 3, 2005 archive of the webpage for “NIST Special Publications”); <https://bit.ly/3P9wM0x> (August 31, 2005 archive of NIST’s homepage – <http://csrc.nist.gov> – which had a link to the “Security Guidelines” page for NIST’s Special Publications, and a direct link to copies of EX1106). Based on the above evidence and my review of the indicia appearing on EX1106 and the U.S. Government’s NIST website stating when EX1106 was published, it is my opinion that EX1106 was publicly accessible in the prior art by being disseminated or otherwise made available to POSAs exercising reasonable diligence by August / September 2005.

173. EX1106 discusses “Symmetric key algorithms” and states that the symmetric “key **shall not** be disclosed to entities that are not authorized access to the data protected by that algorithm.” EX1106, 34 (emphasis original); *see also* EX1106, 35-36 (discussing various “Symmetric Key Algorithms used for Encryption and Decryption,” including AES and “Triple DEA (TDEA)” (aka Triple DES)). As discussed above, Orsini’s obfuscation step violates best practices in the field of securing data, as recognized by NIST, because Orsini exposes a plaintext share key to anyone that obtains an encrypted share. EX1106 at 61 further notes that “[t]he security strength of an algorithm for a given key size is traditional described in terms of the amount of work it takes to try all keys for a symmetric algorithm.” EX1106, 61. As discussed above, with respect to Orsini’s obfuscation, the amount of time it would take to reverse the obfuscation step would have been well-within the capabilities of a determined attacker. (As discussed below, Dr. Rubin admits, Orsini’s obfuscation is “not really involving a lot of math,” Rubin-Depo., 154)

174. And if the attacker knew that Orsini was performing obfuscation by pairing encrypted shares with unencrypted keys (e.g., something that would likely be hard to prevent a rogue insider from discovering), reversing Orsini’s obfuscation step would have been even easier for an attacker to reverse. *See, e.g.*, EX1044, [0002] (noting as background information that while there has been a

“great deal of focus” on “improving computer security” over the Internet, there was a need to “address the problem of providing security once someone has accessed the computer,” like when “many employees may have access to a computer system” and thus there was still a need “to provide a level of security for information on the computer system”).

175. The same would not have been true with a conventional KEK: even if the attacker knew a KEK was used to protect Orsini’s share keys, it would have still been extremely challenging (and possibly beyond a conventional attacker’s skill level) to break the encryption of a KEK. EX1106, 123 (“Cryptographic integrity mechanisms...protect against an active, intelligent attacker.”).

176. EX1106 further directs POSAs that the “*confidentiality* of all symmetric and private keys **shall** be protected.” EX1106, 72 (all emphasis original). It further states that “[w]hen the symmetric or private key exists external to the cryptomodule, confidentiality protection **shall be provided either by encryption (e.g., key wrapping)** or by controlling access to the key via physical means (e.g. storing the keying material in a safe with limited access).” EX1106, 72 (emphasis of “shall” original).¹² Orsini’s method stores the share

¹² Because Orsini is electronically storing key information with the data, using physical security (like storing the plain text keys in a safe) is not compatible with

encryption keys electronically with the data, and so storing the keying material physically in a safe would be contrary to Orsini's system. Thus, EX1106 informs POSAs (as they already knew and consistent with the opinions I offered in my First Declaration), to have "confidentiality protection" of the share keys "provided [] by encryption (e.g., key wrapping)" (i.e., a KEK). EX1106, 72.

177. My opinion that POSAs would have recognized that Orsini's non-obfuscation step would not have provided anything close to the security provided by the cryptographic security provided by a conventional KEK is consistent with testimony Dr. Rubin gave at deposition, as I discuss below.

178. At deposition, Dr. Rubin testified that figuring out the different combinations of encrypted shares and share keys in Orsini's obfuscation step "is a pretty simple math problem" such that "it wouldn't take long" just to write out the combinations by hand. Rubin-Depo., 22-25. Dr. Rubin further testified that in his opinion he didn't "even **think it's really involving a lot of math**" because "[y]ou're just putting something with other things and not putting it with other things..." Rubin-Depo., 154-156. Dr. Rubin further testified that POSAs knew "**you should put your security in the keys** and not in the hiding of your

Orsini's system (let alone a practical solution for most electronic communication and storage systems).

algorithms” but “performing an **obfuscation is [] not putting the protection in the keys**, so I would say **this does not on its own provide security.**” Rubin-Depo., 154-156. I agree that Orsini’s obfuscation step is not putting the security of the encrypted shares in a key (i.e., not encrypting the keys) and that the obfuscation does not on its own provide much (if any) security to prevent an encrypted share from being decrypted.

179. As demonstrated in my First Declaration (§§121-122) and reiterated above, POSAs were well aware that it was undesirable to leave an encryption key unencrypted (as Orsini does with its share key) because it unnecessarily exposed the key (and thus the data it unlocks, in this case the encrypted share) to the risk of a successful attack that could (and POSAs knew would) have been substantially reduced by using the known, conventional cryptographic protection of a KEK. *See, e.g.*, First Declaration, §§121-122.

180. Therefore, as I opined in my First Declaration (§§121-123), encrypting Orsini’s share keys with a KEK would have been obvious to POSAs because it would have been using a known technique (encrypting encryption keys for storage) to solve a known problem (securing the encryption key used to encrypt a piece of data). Doing so required only ordinary skill and there would not have been any unexpected results because using a KEK involved precise mathematics, and, therefore, POSAs would have had a reasonable expectation of success.

1. My KEK-based Obviousness Opinion was Independent of, and based on independent corroborating evidence from, my opinion that Orsini's public-key encryption disclosures provided an anticipating embodiment.

181. Though I continue to believe that Orsini's public-key disclosures are included in a single embodiment that contains Orsini's encryption method disclosed in Orsini's Figure 21, my opinions based on Orsini's public-key encryption disclosures are separate from and were supported by different evidence than my opinions of the obviousness of using a conventional KEK to encrypt Orsini's share keys.

182. Completely independent of Orsini's disclosures about encrypting data to and at data depositories using public-key encryption, my opinion was (and is) that it would have been obvious to POSAs to encrypt Orsini's share keys with a conventionally stored KEK. That obviousness opinion was based on different reasoning and different evidence than any of my opinions regarding anticipation by Orsini. My opinion that Orsini was anticipated by Orsini (in several ways) was premised on Orsini's express disclosures alone, as that disclosure would have been understood by POSAs.

183. However, as I discussed above (§III.A), it is undisputed that Orsini's embodiments, in particular the ones based on Figure 21's method, do not encrypt Orsini's share keys. Thus, I did not opine that Orsini anticipates by encrypting Figure 21's share keys with another key. Instead, I opined that it would have been

obvious for POSAs to do so, using a conventionally stored KEK. As I discussed above (§III.A) and further below (§§III.A.2-III.A.3), the opinion in my First Declaration on the obviousness of using a KEK was based on, for example, POSAs’ background knowledge (which I corroborated with independent evidence) that encryption keys—like Orsini’s share keys—“**should never**” (EX1035, 12:15-23) be transmitted or stored in unencrypted fashion. That evidence – not Orsini’s public-key disclosures – was the corroborating support for my opinion that POSAs had reason to encrypt Orsini’s share keys with a conventional KEK. *Compare* First Declaration, ¶120, ¶124 (discussing public-key encryption of already-encrypted data (**not** the share keys)) *with* First Declaration, ¶¶121-122 (discussing my obviousness opinions based on a conventional KEK encryption to encrypt Orsini’s share keys).

2. My KEK-based Obviousness Opinions Did Not Rely on Common Sense to Supply a Missing Limitation in the Prior Art.

184. I understand that SFI argues that the Petition’s KEK-based obviousness argument relied on common sense to supply a limitation ([1E]) that is allegedly missing in the prior art. POR, 18-32. As I discuss below, SFI’s argument fails on (at least) two accounts: (1) POSAs would not agree that limitation [1E] is missing in the prior art (as demonstrated by multiple corroborating prior art references) and (2) my opinions demonstrating the

obviousness of encrypting Orsini's share keys with a KEK did not rely on POSAs' bare common sense.

a. POSAs would not have unagreed that limitation [1E] was missing in the prior art.

185. As an initial matter, in my opinion, POSAa would not agree that limitation [1E] is missing in the prior art. As I discussed above (*supra* §III.A.1), my opinions on the obviousness of using a KEK were supported by (at least) EX1035 and EX1036, both of which disclose the known desirability of using a KEK to encrypt an encryption key for storage. First Declaration, ¶¶121-122.

186. Moreover, as I discussed in my First Declaration (¶¶196-198), Foster is another example of prior art discloses limitation [1E]. Indeed, SFI and Dr. Rubin do not dispute that Foster's content units are individually encrypted with title keys that are, in turn, encrypted with a KEK.

187. My First Declaration also identified several other disclosures in Orsini and / or obvious variations thereof using well-known encryption techniques that teach and / or render obvious limitation [1E] under Google's and/or SFI's constructions. Though SFI and Dr. Rubin dispute whether POSAs had reason to include those other conventional encryption techniques in Orsini (e.g., cryptographically reordering the data within each encrypted share), SFI and Dr. Rubin do not dispute that those were known encryption techniques that would meet limitation [1E] if performed in Orsini. Thus, those other known encryption

techniques are further evidence of limitation [1E] being known in the art under one or more of the parties' claim constructions.

188. That limitation [1E] is not missing in the prior art is further corroborated by additional prior art I discuss below.

189. In response to SFI's and Dr. Rubin's arguments that limitation [1E] is missing in the prior art, I cite EX1088. EX1088 at 2:20-3:16 teaches to use a KEK to encrypt keys that were used to encrypt "partitions[s]" of "digital content."

190. In response to SFI's and Dr. Rubin's arguments that limitation [1E] is missing in the prior art, I cite EX1080. EX1080 at 2-7 describes a system wherein "[d]ata is encrypted at the source by the TEK" and the system "distribute[s] the key to each receiver." EX1080 further disclosed that the TEK "**must itself be encrypted**" using a KEK. EX1080 at 6.

191. In response to SFI's and Dr. Rubin's arguments that limitation [1E] is missing in the prior art, I cite again EX1106. As I discussed above (§III.A, §III.A.1), EX1106 was published by the US Department of Commerce as a guidelines for best practices in the field of securing data, and it taught that when a symmetric encryption key is used (like Orsini's share keys), and those key "exist[] external to the cryptomodule" (as in Orsini) "confidentiality protection **shall be provided either by encryption (e.g., key wrapping).**" EX1106, 72 (emphasis of "shall" original).

192. As I discussed above, Orsini itself says the encryption keys in its system are generated according to several standardized encryption techniques, including DES and AES. Orsini, [0378]. As I discussed above with corroboration (§III.A.1), both DES and AES disclosed using key wrappers (KEKs) to encrypt their encryption keys. *Infra* §III.B (citing additional corroboration). Thus, POSAs’ knowledge of the very same encryption techniques that Orsini explicitly identifies would have included knowledge of using a KEK encryption key to encrypt the very keys Orsini uses (e.g., using an AES key-wrapping key to protect Orsini’s share keys based on AES encryption).

193. Thus, in my opinion POSAs would disagree with SFI and Dr. Rubin that limitation [1E] is missing from the prior art. Instead, as I demonstrated in my First Declaration and above in response to SFI’s POR, limitation [1E] was not only in the prior art, it was long-known by POSAs in the prior art, and had even been incorporated into the same standardized encryption techniques that Orsini relies on.

b. My opinions on the obviousness of encrypting Orsini’s share keys with a KEK was not based on common sense.

194. But even if it were true that limitation [1E] were not disclosed in the prior art, my opinions in my First Declaration on the obviousness of encrypting Orsini’s share keys did not rely only POSAs’ “common sense.” Rather, I cited multiple corroborating references (e.g., EX1035 and EX1036) to support my

testimony that explained in detail why POSAs had reason to further secure Orsini's shares by using the well-known (not SFI-invented) technique of encrypting encryption keys using conventional KEKs. First Declaration, ¶¶121-122.

195. In response to SFI's argument that limitation [1E] is missing in the prior art, I have cited and discussed additional prior art references above (§III.A.1) that refute SFI's argument. Similarly, in response to SFI's and Dr. Rubin's argument that POSAs lacked motivation to encrypt Orsini's share keys with a KEK, I cited several additional corroborating references in this Reply declaration. *See infra* §III.A.3; *supra* §III.A.1, §III.A.2.a.

196. Thus, my obviousness opinions were and are not based on common sense, but are instead based on my (corroborated) opinion that limitation [1E] was not only in the prior art, but was also within POSAs' general knowledge (corroborated by evidence) of how to secure encryption keys that are used to secure stored and/or transmitted data. Specifically, as I discussed in my First Declaration (¶¶121-123) and further below in response to SFI's and Dr. Rubin's arguments, POSAs would have wanted to encrypt Orsini's share keys because encrypting keys was a known solution the problem how to secure encryption keys stored with encrypted data **and** Orsini's obfuscation step, to the extent it addressed that problem, would have been recognized as being less secure than using a conventional KEK.

197. Thus, POSAs' motivation to use a KEK in Orsini that I have identified is not based on "common sense," but is based on well-understood, documented principles in the field of securing data to encrypt plaintext encryption keys (like Orsini's) with a KEK.

3. My reply to SFI's Challenges to my opinions regarding POSAs' Motivation-to-Combine Orsini with a conventional KEK

198. In its reply, SFI offers three basic challenges to the identified motivations to combine Orsini with a conventional KEK: (1) SFI argues that Orsini already provides sufficient security via its obfuscation step and thus POSAs had no reason to add additional security by encrypting Orsini's share keys with a KEK, (2) SFI argues that encrypting Orsini's share keys with a KEK would have introduced complications that would have dissuaded POSAs from pursuing the combination, and (3) SFI argues the KEK-based obviousness ground is hindsight driven. Consistent with the opinions offered in my First Declaration (§§121-122) and discussed further below, in my opinion POSAs would have disagreed with each of SFI's arguments.

a. It is still my opinion that POSAs Would Have Wanted to Make Orsini's System More Secure by using a KEK.

199. In its POR, SFI argues that POSAs would have had no reason to further secure Orsini's system because (according to SFI) Orsini itself was satisfied

with the security its system provided. POR, 22 (“It is not obvious to modify a reference that already achieves the benefit supposedly served by the proposed modification.”). In my opinion, SFI’s argument misunderstands how POSAs would read Orsini, based on what Orsini itself says and further in light of POSAs’ background knowledge of cryptography techniques and the field of securing data.

200. I first note that the fact that Orsini’s used non-cryptographic obfuscation after encrypting the original data set and each individual share confirms that Orsini itself recognized that there was still a need to further secure Orsini’s encrypted shares. That is, Orsini itself wanted to secure the encrypted shares further. The way Orsini did so was by making it (slightly) harder to access the share encryption keys, by not storing a share key with the share that it encrypted.

201. Orsini **also** discloses that “certain steps of” Figure 21’s method “may be performed in different order, or repeated multiple times, as desired.” Orsini, [0301]. Orsini further discloses in connection with Figure 21’s method that “[e]ach portion of parsed data may be uniquely secured in any desirable way provided only that the data may be reassembled, reconstituted, reformed, decrypted or restored to its original or other usable form.” Orsini, [0301]. Orsini would not have disclosed multiple alterations to Figure 21’s method that added varying forms of security if Orsini (let alone POSAs) actually thought Figure 21’s method was

already “sufficient” (POR, 23), let alone sufficient for all types of data sets, no matter how sensitive the data set being secured.

202. If SFI and Dr. Rubin were correct that Orsini believed that the method illustrated in Figure 21 was “sufficient” (POR, 22-23), then Orsini would not have **repeatedly** identified several **additional** steps that **could** be layered on top of Figure 21’s method to provide **additional** security (e.g., repeating a Figure 21, uniquely securing each share, etc.) That is, rather than teach any of its disclosed methods were sufficient, POSAs would have recognized that Orsini itself recognized that its obfuscation – splitting encrypted portions of data and storing those encrypted portions of data on different storage devices – was potentially insufficient security. POSAs would have agreed that additional (cryptographic) security was often desirable.

203. But rather than use the well-known, pre-existing and conventional cryptographic-based approach of using a KEK, Orsini elected to perform what it termed obfuscation. Even if Orsini thought otherwise (as SFI alleges), POSAs would have recognized that Orsini’s non-cryptographic obfuscation (unlike KEK-based security) provided little additional security over even having no obfuscation at all for a determined attacker. *See also supra* §III.A.1-2. That is, POSAs would have recognized that Orsini’s obfuscation provided **some** additional security, under **some** circumstances (e.g., when an attacker did not possess all encrypted shares),

but using a KEK encryption would have provided (at least) the same security in some cases and additional security in other cases that obfuscation did not provide (e.g., even when an attacker **did** possess all encrypted shares). See EX1100, [0006]-[0011]; [0010] (“Encryption’s purpose is to ensure privacy by keeping information hidden from anyone for whom it is not intended, *even those who have access to the encrypted data.*”).

204. As I discussed in my First Declaration (§§110-123) demonstrating all the obvious methods that POSAs would have had motivation to perform to further secure Orsini’s system and above (§III.A.1-2), even if Orsini had taught or suggested that Figure 21’s method was secure on its own, POSAs would not have agreed. POSAs knew that Orsini’s non-cryptographic technique of storing the share keys separately from the shares they encrypted was **less secure** than doing so while **also** (or alternatively) encrypting the share keys using a KEK. That is because, as I discussed above, unlike using a KEK, Orsini’s obfuscation provided little-to-no additional protection once all of the encrypted shares are collected. *Supra* §III.A (*citing, e.g.,* EX1089, [0569]; EX1090, 2:39-41 (“The more difficult it is for an unauthorized entity to obtain a valid key, the more secure the cryptographic scheme.”)).

205. SFI and Dr. Rubin do not dispute my opinion in my First Declaration that using a KEK would have provided additional security in Orsini’s method. SFI

and Dr. Rubin also do not dispute that cryptographic security would have been almost always (and certainly could have been made) stronger than Orsini's non-cryptographic obfuscation (e.g., by using the encryption techniques available at the time of the invention that were known to be, or at least believed to be, unbreakable in any reasonable amount of time with then-existing technologies).

206. As I discussed above (§III.A), Orsini's Figure 21 divides data into four shares/chunks and stores the share encryption keys with those shares/chunks. There are (at most) only twenty-four different chunk-key pairings that are possible when there are four chunks. POSAs would have understood that computers could easily overcome this obfuscation using a conventional brute-force attack that tried all twenty-four possibilities until the correct one is identified.

207. For example, SFI's EX2007 (at 11) discloses that even "26 possible keys" is subject to "a *brute-force attack*" but "any secure encryption **must not be vulnerable to such brute-force attack**" (italicized emphasis original). In the same way, Orsini's 24 different obfuscation possibilities would be subject to a conventional brute-force attack that would have been more vulnerable than using the conventional cryptographic-based security of a KEK.

208. Moreover, as I discussed above (§III.A), neither Orsini nor the patent's claims required four shares and four share keys, but rather allow for as few as two shares and two share keys, which has only **one** way to non-

cryptographically obfuscate by swapping one share's key for the other, as Dr. Rubin agreed at deposition (Rubin-Depo., 24-25). Plainly, only a single way to obfuscate is not providing the same level of protection that conventional encryption techniques with a KEK would have provided by the mid-2000s.

209. Thus, POSAs would have recognized that SFI's assertion that Orsini's non-cryptographic obfuscation "achieves the benefit...served by" a KEK's cryptographic protection (POR, 23) is incorrect. Rather, POSAs would have understood that using a KEK would have provided independent, and almost certainly in every case (but at least in many cases) better, security than Orsini's obfuscation, whether both were used together or only KEK-based security was used in place of Orsini's non-cryptographic obfuscation.

210. SFI alleges that Orsini teaches that its obfuscation was "sufficient" to protect its data. POR, 24 (citing Orsini, [0353]). As I discussed above, in my opinion, those boilerplate statements in Orsini would not have in any way discouraged POSAs from modifying Orsini to use better and/or alternative cryptographic techniques (like a KEK) to further increase security against unauthorized attack. Indeed, the '609 patent, which allegedly recognized the inadequacies of Orsini's technique and allegedly solved it by using a KEK (a well-known technique) for its well-known purpose, has the same disclosures as Orsini in that regard. *Compare* Orsini, [0353] *with* EX1001, 60:17-20.

211. But even if SFI were right that Orsini itself thought that obfuscation was “sufficient” (POR, 23), that would not have made using the known KEK-based cryptographic technique non-obvious to POSAs to use in Orsini’s system as an alternative and/or additional (and more-secure) way to add security, and *further secure the data chunks* (under SFI’s view of how that limitation can be met). I have been informed that POSAs are hypothetical people that are presumed to be aware of all the pertinent prior art. As I discussed above (§III.A.1-2), my First Declaration and this reply have cited numerous prior art references in the field of securing data that unambiguously taught the benefits of and the risks in not encrypting encryption keys with a conventional KEK.

212. POSAs would have known that even when a prior art reference (like Orsini) discloses a certain level of a solution to a problem (e.g., Orsini’s use of obfuscation to provide some additional security against discovering encryption keys and decrypting shares), POSAs in the field of securing data would have sought out another known (and in my opinion would have been recognized as an obviously more secure) solution to the same problem (in this case, using a KEK to provide meaningful cryptographic protection against discovering Orsini’s share keys). That is especially true to POSAs in the field of securing data because POSAs knew “[t]he more difficult it is for an unauthorized entity to obtain a valid key, the more secure the cryptographic scheme.” EX1090, 2:39-41. POSAs would

have recognized that for (at least) high importance *data sets* and/or risk levels, additional security would have been desired and worth whatever complexity and/or cost using a KEK would have required. And given the ubiquitousness of using KEKs in the prior art, as I've demonstrated in my First Declaration and above, POSAs would have found the complexity and/or cost to be negligible or at least resolvable using their ordinary skill with a reasonable expectation of success.

b. In my opinion, any alleged Drawbacks to the modifying Orsini to use a KEK that SFI Identifies would have been Easily Overcome

213. SFI argues that alleged “drawbacks associated with managing and securing” KEKs would have dissuaded POSAs from using KEKs in Orsini. POR, 23-29. As I explained in my First Declaration (¶¶121-122) and in response to SFI’s arguments, it is my opinion that POSAs would not have been dissuaded from using KEKs in Orsini. Instead, POSAs would have had clear reason to use a KEK in Orsini and a reasonable expectation of success in doing so. I also note that SFI does not allege that addressing these alleged “drawbacks” was beyond POSAs’ skill. As I discuss below, these alleged drawbacks would not have been insurmountable, but instead would have required only ordinary skill to consider and deal with. That is why POSAs would have had (at least) a reasonable expectation of success in implementing a KEK in Orsini.

214. I note first that Orsini **already** taught techniques for managing and securing KEKs, and certainly could have been modified in a conventional, straightforward way to use conventional techniques for doing so. For example, as I discussed above (*supra* §III.A), Orsini contemplates using and supports “Triple DES and AES” (Orsini, [0378]), which already taught the use and management of KEKs, as I demonstrated above with corroborating evidence.

215. In addition, Orsini also **explicitly** says it supports “a wide number of alternative key management systems.” Orsini, [0107]. Several prior art references I cited in my First Declaration (and additional corroborating evidence I cite in response to SFI’s reply arguments) demonstrate POSAs knew that using and managing (e.g., creating and storing) KEKs was conventional by the time of the patent’s alleged priority date.

216. In response to SFI’s arguments and Dr. Rubin’s opinions challenging POSAs’ expectation of success in modifying Orsini to use and manage KEKs I cited EX1036 (cited First Declaration, ¶¶121-122). EX1036 discloses a “system for **managing keys**,” including KEKs (2:29-61) and says the “prior art” already included “security infrastructure to solve the key management problem[]” (2:7-22).

217. In response to SFI’s arguments and Dr. Rubin’s opinions challenging POSAs’ expectation of success in modifying Orsini to use and manage KEKs, I further cite SFI’s EX2032. EX2032 discloses that IBM had already “designed and

developed” a commercialized “Common Cryptographic Architecture” “to provide” for, for example, “key management.” EX2032, 683. Among those keys that were managed by that prior art (commercial) system were “key-encrypting keys (KEKs)” that were “used to export and import encrypted keys between systems.” EX2032, 683. EX2032 further discloses “encrypt[ing] session keys using [a] master key.” EX2032, 683-684.

218. In response to SFI’s arguments and Dr. Rubin’s opinions challenging POSAs’ expectation of success in modifying Orsini to use and manage KEKs, I further cite EX1017 (cited First Declaration, ¶85). EX1017 (at 1:50-1:68) discloses as background information that that “[t]here are generally two categories of encryption key variables in a secure communication system.... The second type is called **a key encryption key (KEK), and is used to encrypt encryption key variables for transit, transmission or storage.**” *See also* First Declaration, ¶85 (discussing EX1017). EX1017 further discloses as background information that “[i]n more advanced secure communication systems, encryption **key variables are stored and distributed electronically to communication units using a key management controller or KMC.**” EX1017, 1:50-1:68. That is, it was background knowledge as of 1993 how to store, distribute, manage, and use KEKs for secure storage.

219. In response to SFI's arguments and Dr. Rubin's opinions challenging POSAs' expectation of success in modifying Orsini to use and manage KEKs, I further cite EX1106. As I discussed above (*e.g.*, §III.A), EX1106 is a document published by the U.S. Department of Commerce's National Institute of Standards and Technology ("NIST"), titled "Recommendation **for Key Management** – Part 1: General." This document provided "basic key management guidance" to advise on the "best practices' associated with key management." EX1106, 5. The various key-management principles discussed in EX1106 include for KEKs for the standardized encryption techniques discussed in Orsini, like AES and DES. EX1106, 35-36.

220. Thus, the above corroborating evidence supports my opinion in my First Declaration that any potential drawbacks of using having to manage KEKs were not beyond POSAs' skill and thus would not have discouraged POSAs from pursuing the use of a conventional KEK to secure Orsini's share keys.

221. But even if the alleged "drawbacks" to KEKs that SFI identifies were true, that would still not defeat have dissuaded POSAs from using this known technique. SFI does not dispute that using the known KEK technique in Orsini's system would have increased security because Orsini stores its share keys outside of Orsini's encryption device in unencrypted form, which the prior art taught

“should **never** occur” for security reasons, as I discussed in my First Declaration (§§121-122) and above (§III.A-III.A.3.a). *E.g.*, EX1035, 12:15-23.

222. Indeed, POSAs would have known that securing data often involved tradeoffs in “convenience and security, speed,” etc. EX1081, 15:6-25. For example, for particularly if data to be secured were of high value, POSAs would have known that any (marginal) increase in cost or complexity of using the cryptographic protection of a KEK would have been desirable nonetheless given the (relatively weak) additional protection that Orsini’s non-cryptographic obfuscation would have provided. Thus, even if using KEKs to encryption Orsini’s share keys would have added some complexity (and I don’t believe it would have given the already-existing structures in Orsini’s system) that still would not have dissuaded POSAs from pursuing the use of a KEK in Orsini to secure its share key. That is because using and managing KEKs were well-known to POSAs and were known to have practical application in a system (like Orsini’s) that stores and transmits encryption keys.

223. Thus, in my opinion, encrypting Orsini’s share keys would not have been beyond POSAs’ capability, at least because it was nothing more than using a known technique (KEK-based encryption of keys) to address a known problem (the security risk of storing an unencrypted encryption key outside the cryptographic device), using prior art elements of KEK-based encryption

according to their established function. Moreover, as I discussed in my First Declaration (¶¶121-122) and above (*e.g.*, §III.A), using the cryptographic protection of a KEK would have obtained security benefits that were not possible from Orsini's Figure 21 in the absence of using a KEK.

c. My opinion KEK-based Obviousness Opinion was not based on hindsight, but was instead grounded in evidence that I cited.

224. SFI contends that the Petition's KEK-based obviousness argument is driven by hindsight. POR, 28-29. To the extent SFI is contending that my opinions were also driven by hindsight, I do not agree that my opinions on the obviousness of using a KEK in Orsini's system to encrypt Orsini's share keys was based on hindsight. Instead, the combination of Orsini with conventional KEK-based protection that I discussed in my First Declaration (¶¶121-122) and further herein in response to SFI's argument, is based on a specific motivation—that I supported by multiple *corroborating evidence*—to use a KEK for its well-known purpose to avoid the well-known (and unnecessary and avoidable) risk of having Orsini's share keys stored outside the cryptographic device in non-encrypted form. *See* First Declaration, ¶132 (citing **EX1035**, 12:15-23, 10:42-62; **EX1036**, 1:66-2:6; *supra* §III.A-III.3.b (citing additional corroborating evidence in response to SFI's arguments and Dr. Rubin's opinion demonstrating that POASs would have

recognized based on already-existing principles and techniques the benefits of using a KEK to secure plaintext keys like Orsini's share keys).

225. Again, Orsini itself recognized a need to protect its share keys. Orsini elected to add some protection for its share keys by using a non-cryptographic obfuscation step, rather a conventional KEK technique. However, POSAs would have recognized that an alternative (or additional) and conventional well-known way to protect encryption keys like Orsini's share keys would have been to use the cryptographic protection provided by a KEK. As I've discussed above (§III.A), POSAs would have recognized that using a KEK would have provided at least as much (and in many situations **more**) security as Orsini's non-cryptographic obfuscation. POSAs would have recognized that even in an implementation that continued to use non-cryptographic obfuscation, adding the cryptographic protection of a KEK would have provided additional security. Indeed, the alleged invention of the '609 is patent over Orsini is to further secure the shares with an encryption operation based on an encryption key. If that limitation can be met by encrypting the share keys with a KEK (as SFI's infringement contentions allege), my First Declaration and this Reply Declaration have demonstrated with **multiple pieces of corroborating evidence** that using a KEK was already well-known by POSAs to provide the same additional security that SFI alleges its patent added over Orsini.

B. Response to Argument that POSAs would not have a reasonable expectation of success in modifying Orsini to meet limitation [1E]

226. In its POR, SFI asserts that POSAs would not have had a “reasonable expectation of success in modifying Orsini to achieve claim 1.” POR, 4. Though SFI makes that broad statement for all of the different obvious implementations of Orsini that meet the challenged claims, I note that SFI never identifies any specific reason **why** POSAs allegedly would not have had a reasonable expectation of success in using a conventional external KEK to encrypt encryption keys (like Orsini’s), as was known in the art. SFI cites no independent evidence (beyond Dr. Rubin’s opinions) to substantiate SFI’s unexplained reasons why POSAs would not have reasonably expected success in modifying Orsini to use a conventional KEK to encrypt Orsini’s share keys.

227. As I opined in my First Declaration, POSAs would have reasonably expected success in using a KEK in Orsini because doing was “conventional and well-known to POSAs” and because Orsini itself says that additional steps may be performed. First Declaration, ¶¶106-107. That opinion was backed by the multiple corroborating references I cited that demonstrated that using and managing KEKs was well known and conventional in the prior art. First Declaration, ¶54, ¶¶68-69, ¶85, ¶¶121-122 (citing EX1035, EX1036).

228. In response to SFI’s argument that POSAs would not have reasonably expected success in using a KEK in Orsini, I note again that Orsini itself uses “Triple DES and AES” encryption techniques to encrypt the shares. Orsini, [0378]. As I discussed above (§III.A), these were standardized encryption techniques that taught encrypting keys using KEKs to secure those keys for storage. *See* EX1094, [0056]-[0057] (“Triple-DES Key Wrap” (i.e., KEK)); EX1090, 19:47-61, 21:1-22:54 (noting “the National Institute of Standards and Technology (NIST) Advanced Encryption Standard (AES)” provides a “key wrapping function” (i.e., a KEK function)).

229. In response to SFI’s argument about POSAs expectation of success, my continuing opinion that POSAs would have reasonably expected success in using, managing, and storing KEKs in Orsini because it was a well-known, conventional technique is demonstrated by additional corroborating evidence.

230. In response to SFI’s argument that POSAs would not have reasonably expected success in using a KEK in Orsini, I cite EX1017. EX1017 (at 1:50-68) (also discussed *supra* §III.A) discloses as background information that “**a key encryption key (KEK), and is used to encrypt encryption key variables for transit, transmission or storage**” and that “[i]n more advanced secure communication systems, encryption **key variables are stored and distributed**

electronically to communication units using a key management controller or KMC.”

231. In response to SFI’s argument that POSAs would not have reasonably expected success in using a KEK in Orsini, I further cite EX1093. EX1093 (at [0013]) discloses an example of using “secure key management when separate devices are used for generating and utilizing the keys” (e.g., externally retrieved keys), and as part of that prior art key management there would be “a key manager that facilitates exchange of the KEK used to encrypt and decrypt the keys generated by the host processor.” *See also* EX1093, [0035]-[0039] (similar), [0057]-[0060].

232. In response to SFI’s argument that POSAs would not have reasonably expected success in using a KEK in Orsini, I further cite EX1095. EX1095 (at 1:32-47) discloses that it was POSAs’ “BACKGROUND:” to have “[c]ryptographic key managements schemes [that] are built on **commonly used**...concepts, such as key wrapping” (i.e., KEKs). EX1095 (at 3:10-15) recognizes that “KEK[s] allows keys to be exported **for storage**.”

233. Thus, consistent with my First Declaration, numerous prior art references corroborate my opinion that POSAs would have had a reasonable expectation of success in using a KEK in Orsini’s system to encrypt Orsini’s share keys. Indeed, using a KEK was based on deterministic mathematical processes

(not guess work) that would have further bolstered POSAs' reasonable expectation of success.

234. In his reply declaration, Dr. Rubin asserts that Orsini's "**session** key management is very different than key management for longer-lived keys such as the other 'conventional encryption keys' Petitioner relies upon." EX2033, ¶73. Dr. Rubin asserts that the "longer lifespan" of longer-lived keys "makes potential compromise of a key-encryption key all the more consequential" and that a "key management solution for session keys would not be appropriate for the 'group key,' 'encryption key,' or 'key-encryption key'" and would require "necessarily" "different" "key management" than for the "session" keys. EX2033, ¶¶79-80. I disagree and in my opinion POSAs would disagree. As I discussed above, POSAs would have been well-aware of numerous solutions for key-management systems for a wide-variety of encryption keys, including conventional key-encryption keys (KEKs). As I discussed above (with corroboration), Orsini itself discloses using pre-existing encryption standards, such as AES and DES, which **already** included key management solutions for their encryption keys, which included KEK.

235. Moreover, one of Orsini's keys is a "Parser Master Key," which Orsini says is "an individual key associated with the installation of the data parser" that "is installed on the server" and "secur[ed]" in various ways including "separate hardware key store, standard key stores, customer key stores or within a

secured database table.” Orsini, [0399]-[0400]. That is, the Parser Master Key is an example of what Dr. Rubin is alleging is a longer-lived key and Orsini discloses that it **already** relies on existing, conventional key management systems to manage that key. POSAs would have known that a KEK could have (at least) been managed using the same systems. Orsini similarly notes that there are “a wide number of alternative key management systems,” including those for “public-key system[s].” Orsini, [0107]. POSAs knew that public-private encryption key pairs were (and are) intended to be long-lived keys and that the key-management of public-key encryption systems was substantially more challenging than the management of a conventional, externally retrieved and stored (symmetric) KEK. But as even Orsini recognizes, the management of public-encryption keys had already been solved in the prior art. As I demonstrated above with corroborating evidence, the same was true of conventional KEKs like the ones I opined would have been used in an obvious modification of Orsini.

236. Moreover, Dr. Rubin did not identify where the ’609 patent discusses any of the alleged challenges in (let alone provide a solution to) key-management problems for a key-encryption key, or any of the ’609 patent’s alleged external keys. That reinforces to POSAs that the ’609 patent is relying on **existing** key-management systems and techniques for its external key, thus reinforcing their expectation of success in using the same or similar available key management

systems for a KEK. In fact, Dr. Rubin testified at deposition that one of the patent's disclosed external keys is the session master key: "I agree that the session key is an external key." Rubin-Depo., 28. That admission contradicts his declaration opinion that Orsini's sessions keys are "very different than key management for longer-lived keys" (EX2033, ¶78) because at deposition he admitted that Orsini is **already** disclosing examples of an external key, and thus management thereof, and SFI's infringement contentions have mapped the external key to a KEK encryption key. *See, e.g.*, EX 1071, pdf. 47. That is, to the extent the '609 patent has described a key-management system for an *external key* that could be a KEK, POSAs would understand the same would have been true of Orsini, as I see no meaningful, additional disclosures of key management for external keys in the '609 patent compared to Orsini.

237. (I note, for clarity, that I have not been asked to consider whether the '609 patent discloses using a KEK in the manner required of the claimed *external key*, but that is a different question from whether POSAs would have known how to manage such keys, whether or not those management systems were explicitly disclosed in Orsini.)

238. To POSAs, the management of a KEK was, from a technical point of view, little different from the management of any other encryption key in Orsini's system. In conventional key management of a KEK, the KEK would have been

retrieved from an external storage system (e.g., a conventional key store from within or from outside the system),¹³ and after being applied to Orsini's share keys, a record would have been kept (likely in an encryption form) about what KEK was used to encrypt those share keys, and then the KEK could have been returned / placed into storage or destroyed, with a record of where the KEK could be retrieved from when needed.

239. Thus, POSAs would not have agreed with Dr. Rubin's opinions on the alleged complexities and differences in management of allegedly shorter-lived sessions keys versus allegedly longer-lived KEK encryptions. Rather, to the extent Orsini did not already have in place key-management systems designed to include management of KEKs (and I believe they would have recognized that Orsini's key-management systems were already designed for that), POSAs would have recognized that it would have required only ordinary skill to do so using well-known, conventional key-management techniques and systems (much like SFI's patents rely on existing key-management solutions).

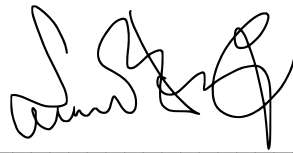
* * *

¹³ When retrieved from an outside system (and possible even when retrieved from an inside system), the KEK would itself be encrypted by another encryption key (e.g., the recipient's public key so that only that recipient could decrypt it using their private key).

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

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

Dated: December 23, 2024

A handwritten signature in black ink, appearing to read 'Samrat Bhattacharjee', written over a horizontal line.

Samrat Bhattacharjee, Ph.D.