

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC and
LENOVO (UNITED STATES) INC.,
Petitioners,

v.

ERICSSON AB,
Patent Owner.

IPR2025-00003
Patent No. 9,313,178

DECLARATION OF ANTHONY J. WECHSELBERGER

Google Exhibit 1003 Google v. Ericsson

TABLE OF CONTENTS

I.	PERSONAL AND PROFESSIONAL BACKGROUND	1
A.	Educational Background	1
B.	Professional Background.....	2
C.	Patents and Publications	8
II.	MATERIALS REVIEWED AND CONSIDERED	8
III.	MY UNDERSTANDING OF PATENT LAW	10
A.	Anticipation	12
B.	Obviousness.....	13
IV.	BACKGROUND	16
A.	'178 Patent.....	16
B.	Prosecution History of the '178 Patent.....	16
V.	PERSON OF ORDINARY SKILL IN THE ART (“POSITA”).....	17
VI.	CLAIM INTERPRETATION	19
VII.	PRIOR-ART REFERENCES.....	20
A.	Peterka	20
B.	Suzuki	28
C.	Bocharov.....	30
D.	Peterka308	34
E.	Chen.....	35
VIII.	THE CHALLENGED CLAIMS ARE UNPATENTABLE IN LIGHT OF THE PRIOR ART IDENTIFIED IN THE PETITION	36
A.	<u>Ground 1</u> : Peterka Renders Claims 1-6, 10, 12, 16-18, and 20 Obvious.....	37
1.	Claim-by-Claim Analysis.....	39
a.	Claim 1	39
i.	[1PRE] “A method for handling secure distribution of content comprising:”	39
ii.	[1A] “initiating a media playback request and receiving a playback request response;”	39

iii.	[1B] “parsing content information from the playback request response, the content information including content encryption keys, content encryption key identifiers, and content encryption key expiration times;”	45
	(1) Content Information	46
a)	Content Encryption Keys.....	46
b)	Content Encryption Key Identifiers.....	49
c)	Content Encryption Key Expiration Times	52
	(2) Peterka Teaches a Playback Request Response Comprising “Content Information” for Multiple Segments.....	55
iv.	[1C] “retrieving content and manifest files from a content delivery server;”	56
v.	[1D] “detecting content encryption key rotation boundaries between periods of use of different content encryption keys in decrypting retrieved content;”	67
vi.	[1E] “issuing requests to a license server ahead of a key rotation boundary to retrieve a second content encryption key to be used after a content encryption key rotation boundary is reached; and”	80
vii.	[1F] “applying the second key for content decryption after the key rotation boundary is reached.”	83
b.	Claim 2: “The method of claim 1, wherein the content encryption keys and content encryption key identifiers returned in the playback request response include the content encryption key and associated identifier currently being applied as well as a futurecontent encryption key and associated identifier yet to be applied.”	85
c.	Claim 3: “The method of claim 1, wherein the content encryption key expiration returned in the playback request response is expressed as an expected minimum interval between periodic key rotation.”	86

d.	Claim 4: “The method of claim 3, further comprising: timing the prefetching of a next un-retrieved content encryption key based on an expected expiration of the content encryption key currently being used.”	90
e.	Claim 5: “The method of claim 4, further comprising: prefetching a next un-retrieved key a fixed duration before the expected expiration of the content encryption key currently being used.”	91
f.	Claim 6: “The method of claim 4, further comprising: prefetching a next un-retrieved key within a period of time after the expected expiration of the content encryption key currently being applied, the period of time beginning at the expected expiration of the content encryption key currently being used, and ending at a duration calculated as a fixed percentage of a fixed periodic content encryption key expiration interval.”	93
g.	Claim 10: “The method of claim 1, wherein license server communications are secured and authenticated using a selected one of Secure Sockets Layer and a token-based technique using a token encrypted with a symmetric key.”	96
h.	Claim 12: “The method of claim 1, wherein actual content encryption key rotation boundaries are detected based on real-time in-band notifications.”	100
i.	Claim 16.....	102
i.	[16PRE] “A computerized device operable as a client for handling secure distribution of content, comprising:”	102
ii.	[16A] “memory operative to store computer program instructions; one or more processors; input/output interface circuitry; and interconnect circuitry coupling the memory, processors and input/output interface circuitry together,”	102
iii.	[16B]-[16G]	103
j.	Claims 17-18, and 20	103
B.	<u>Ground 2</u> : Peterka in View of Suzuki Renders Claims 1-6, 10, 12, 16-18, and 20 Obvious	104

1. Peterka-Suzuki.....	104
2. Claim-by-Claim Analysis.....	110
a. Claim 1-2, 10, 12, 16-18, and 20	110
b. Claims 3-6	114
C. <u>Ground 3</u> : Peterka in View of Bocharov Renders Claims 1-6, 10, 12, 14, 16-18, and 20 Obvious	117
1. Peterka-Bocharov	117
2. Claim-by-Claim Analysis.....	123
a. Claims 1-6, 10, 12, 16-18, and 20	123
b. Claims 12 and 20: “...wherein actual content encryption key rotation boundaries are detected based on real-time in- band notifications.”	126
c. Claim 14: “The method of claim 12, wherein content encryption key rotation boundaries are detected based on notifications present in the content manifest file, the content manifest file being selected from an m3u8 file, an IIS Smooth Streaming manifest fle, a DASH MPD file and a proprietary manifest file.”	130
D. <u>Ground 4</u> : Peterka in View of Suzuki and Bocharov Renders Claims 1-6, 10, 12, 14, 16-18, and 20 Obvious.....	132
1. Peterka-Suzuki-Bocharov.....	132
2. Claim-by-Claim Analysis.....	133
a. Claims 1-6, 10, 12, 14, 16-18, and 20.....	133
E. <u>Ground 5</u> : Peterka in View of Bocharov, Peterka308, and Chen Additionally Renders Claims 7-8 and 19 Obvious.....	135
1. Peterka-Bocharov-Peterka308-Chen.....	135
2. Claim-by-Claim Analysis.....	140
a. Claims 7 and 19: “The [method of claim 1 / computerized device of claim 16] wherein subsequent content encryption key identifiers are predictable based on a predetermined known progression.”	140
b. Claim 8: “The method of claim 7, further comprising: the identifier being calculated as monotonically increasing	

	sequential integer values based on the number of segments or video frames generated during a fixed periodic content encryption key expiration interval.”	142
IX.	CONCLUSION.....	143

I, Anthony J. Wechselberger, declare:

1. I have been retained by Wolf, Greenfield & Sacks, P.C., counsel for Petitioners Google LLC and Lenovo (United States) Inc. (collectively, the “Petitioners”), to assess claims 1-8, 10, 12, 14, and 16-20 (the “Challenged Claims”) of U.S. Patent No. 9,313,178 (“the ’178 Patent”). I am being compensated for my time at my standard consulting rate, plus actual expenses. My compensation is not dependent in any way upon the outcome of the *inter partes* review of the ’178 Patent. I have no other personal or financial stake or interest in the outcome of the present proceeding. I do not have any expectation or promise of additional business with the Petitioners in exchange for the positions explained herein.

2. I make this Declaration based on my personal knowledge, including my education, training, research, and professional experience.

I. PERSONAL AND PROFESSIONAL BACKGROUND

3. I have summarized in this section my educational background, career history and other relevant qualifications.

A. Educational Background

4. I have a bachelor’s degree and a master’s degree in electrical engineering, from the University of Arizona in 1974 and San Diego State University in 1979, respectively. I also completed the Executive Program for Scientists and Engineers at the University of California at San Diego in 1984.

B. Professional Background

5. I am currently the President of Entropy Management Solutions, a position I have held since I founded the company in 1999. In this capacity I perform consulting services related to technology and business development, content management, distribution and merchandizing, systems engineering, and product design in the areas of broadband and multimedia technologies and their associated commercial systems.

6. I have over forty-five years of experience working with high-technology systems related to military, commercial, and consumer communication systems and networks. I have held various design, leadership, and executive positions in, for example, engineering, operations, sales and marketing, and product management at leading companies in those fields, such as TV/COM International, Inc. (“TV/COM”) and Oak Communications, Inc. (“Oak”).

7. One of my areas of specialization is command and control in connection with broadband communications networks. “Command and control” refers to the technical oversight and management of communication systems and equipment within a distribution system to direct the equipment as to its set-up and operation in order to perform required functions. For example, in broadband networks used in the television industry (cable, satellite, broadcast, Internet), command and control typically operates in the background (i.e., invisible to consumers) to enable the

system operator to manage the delivery of content to targeted receivers, such as mobile devices, smart televisions, and set-top boxes (“STB”) according to the subscription services the consumer is authorized to access. Likewise in the case of the ’178 Patent of the current matter, command and control communications operate within a distributed system to implement the claimed security infrastructure features and functions.

8. I also specialize in the area of rights management. “Rights management” controls or oversees the consumption of multimedia content according to defined rules on behalf of the content-rights owner or network operator. As today’s delivery processes have become more digitally enabled, the term “Digital Rights Management” (“DRM”) has been adopted to refer to these processes. Command and control and DRM are of particular relevance to the ’178 Patent of this matter because they are central to the Challenged Claims as forms of metadata that are generated for, and/or accompany the streamed content.

9. In the present matter, the implementation of command and control and DRM solutions involve the use of what are referred to as “in-band” and/or “out-of-band” communication control channels. Such control channels are used for the delivery of command and control and DRM information. My personal experience with in-band and out-of-band control channels for content/multimedia delivery systems and networks dates back to 1980, and my familiarity with the types of

equipment and applications implementing such control channels dates back to the early 1970s. Additionally, in the current matter the combination of command and control and DRM are used with cryptographic technologies such as encryption/decryption and key management strategies, and I have decades of experience with these as well.

10. I also have significant experience with interactive and client-server technologies, such as those used in two-way broadband networks, computer systems and Internet networks. Network management and command and control includes the technical oversight and management of communication systems and equipment within a distribution system to direct both the transmission equipment (*e.g.*, network infrastructures, servers, hubs, nodes, head ends, and uplinks etc.) and receiving equipment (*e.g.*, personal computer, television, set-top box, handset/mobile device, or other consumer appliance) as to communications, applications, and setup and operation, in order to perform required features and functions.

11. As a Director and then Vice President at Oak from 1982-1990, and as Chief Technology Officer at TV/COM from 1990-1995, I was involved in the development of terrestrial broadcast, satellite uplink, and cable headend industrial equipment for television transmissions, as well as consumer appliance equipment such as STB and other home-based or home-networked devices. All of these solutions involved computer control systems for network and associated network-

device command and control, and for management of content distribution and consumer appliance functions.

12. I was an active participant and contributor to the International Organization for Standardization MPEG-2 digital television standards initiatives and the European Digital Video Broadcast (“DVB”) forum (which was based upon MPEG-2). Through my employer TV/COM, I was a voting member of the U.S. Advanced Television Systems Committee forum (which was also based upon MPEG-2). As Chief Technology Officer of TV/COM between 1990-1995, I participated in the development of a business strategy based on supporting open international standards for digital television (“DTV”).

13. As the technologies and standards in support of DTV moved towards implementation in the mid-1990s, the dawn of the Internet age arrived. This had a dramatic impact on the way broadband systems engineers like me began to plan for the future. From a high-level perspective, the composition of DTV is simply “data,” and the concept of convergence—the melding of traditional broadband communications systems and equipment, computers and computer networks, and the telecommunications worlds—allowed those technologies to coalesce. Support for on-line and Internet services demanded a high-performance two-way data transmission capability, and so broadband network providers began to upgrade their distribution infrastructures accordingly.

14. In conjunction with this convergence, as TV/COM's Chief Technology Officer, I directed the expansion of our network products into broadband data communications generally, from its initial focus on digital television. Networks became more advanced in order to support real-time interaction between consumers and information sources within the network. Interactive and on-line applications led to rapid adoption of client-server information-access approaches (typical of the computer industry) in the products and technologies I worked with, for content delivery and network command and control functions. Starting in the mid-1990s, the ubiquitous STB began to evolve from a minimalist appliance towards a communications hub of the consumer's media room. In this same time period, the PC had also become a ubiquitous consumer appliance, and with the Internet age came much innovation in electronic-information distribution and electronic merchandising (i.e., art related to complementing physical information media and brick-and-mortar institutions with all-electronic digital alternatives). TV/COM and I were part of this evolution until TV/COM was purchased in 1999.

15. In my consulting work I have continued to work with technologies and network infrastructures for content distribution and management. My current work involves both traditional and newly developing architectures and distribution channels. As an example of the latter, I am the chief security systems architect on behalf of the five major Hollywood studios for their "Digital Cinema Initiatives"

(“DCI”) consortium. DCI develops and evolves requirements and specifications for transitioning first-run theatrical movie releases from film to digital for distribution and exhibition display. On behalf of the studios, I am responsible for all elements of command and control and DRM for digital cinema system design and implementation. I also represent DCI at the Society of Motion Picture and Television Engineers (“SMPTE”), which has been, and continues, developing the set of internationally recognized standards for global adoption of digital cinema. In this capacity I have been an editor of or contributor to a dozen SMPTE standards, a role I continue today.

16. The migration to all-digital distribution impacts other content distribution channels such as early window release for hospitality, airplane, and cable/satellite video-on-demand, as well as so called “over-the-top” distribution channels based on Internet distribution. I have also been a technology consultant to content management and distribution entities in these areas.

17. My consulting practice today includes a balance of technology and systems-engineering services and assistance to the legal community as a technology consultant and expert witness. I have provided expert testimony in the areas of broadband communications and multimedia technologies, and the associated networks and appliances as used for content management and delivery on many occasions.

C. Patents and Publications

18. I am a named inventor on two patents, U.S. Patent No. 4,531,020, “Multi-layer Encryption System for the Broadcast of Encrypted Information,” and U.S. Patent No. 5,113,440, “Universal Decoder.” I am an inventor on two patent applications: U.S. Patent Application No. 10/028,013, filed December 21, 2001, and entitled “Storage and Delivery of Electronic Media Content with Advertising” (abandoned); and U.S. Patent Application No. 17/499,824, filed October 12, 2021, and entitled “Remote Audience Participation at Live Events” (abandoned). I have participated in U.S. patent prosecution and have a general understanding of the process and of the novelty and non-obviousness requirements for patentability, although I am not a lawyer or patent attorney.

19. Over many years I have published and presented a number of articles and papers related to content/information creation, transmission/distribution and reception/consumption in various media sectors, including cable, satellite, broadcast/wireless, Internet, and digital cinema.

20. My curriculum vitae is provided as Exhibit 1004.

II. MATERIALS REVIEWED AND CONSIDERED

21. My findings, as explained below, are based on my years of education, research, experience, and background in the field of information security systems that employ cryptographic keys to encrypt/decrypt digital data, as well as my

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

Exhibit	Description
1001	U.S. Patent No. 9,313,178 (“178 Patent”)
1002	Prosecution History of U.S. Patent No. 9,313,178 (“178 FH”)
1005	U.S. Patent Publication No. 2002/0172368 (“Peterka”)
1006	Certified Translation of Japanese Published Patent Application No. 2004-166153 (“Suzuki”)
1007	U.S. Patent Publication No. 2010/0235528 (“Bocharov”)
1008	U.S. Patent Publication No. 2008/0270308 (“Peterka308”)
1009	European Patent Publication No. 1,418,756 A2 (“Chen”)
1010	U.S. Patent Publication No. 2005/0138362 (“Kelly”)
1011	U.S. Patent No. 8,677,428 (“Lewis”)
1012	U.S. Patent No. 8,789,196 (“Pestoni”)
1013	International Publication No. WO 2010/107625 A2 (“Sood”)
1014	U.S. Patent No. 8,806,050 (“Chen050”)
1015	T. Stockhammer, “Dynamic adaptive streaming over HTTP: Standards and design principles,” in Proc. ACM MMSys’11, San Jose, CA, USA, Feb. 23–25, 2011, pp. 133–144 (“Stockhammer”)
1016	W. Van Lancker, D. Van Deursen, E. Mannens and R. Van de Walle, “HTTP adaptive streaming with Media Fragment URIs,” 2011 IEEE International Conference on Multimedia and Expo, Barcelona, Spain, 2011, pp. 1-6, doi: 10.1109/ICME.2011.6012149 (“Van Lancker”)
1017	3GPP TS 26.234 version 9.3.0 Release 9 (2010-06), Technical Specification, “Universal Mobile Telecommunications Systems (UMTS); LTE; Transparent end-to-end Packet Streaming Services (PSS); Protocols and codecs,” Reference RTS/TSGS-0426234v930 (June 2010) (“3GPP-AHS”)
1018	R. Pantos (Ed.), HTTP Live Streaming, draft-pantos-http-live-streaming-06 (Mar. 31, 2011) (“Pantos”)
1019	RFC793: Transmission Control Protocol (September 1981) (“RFC793”)
1020	RFC2616: Hypertext Transfer Protocol – HTTP/1.1 (June 1999) (“RFC2616”)

1021	The Secure Sockets Layer Protocol, V. 3.0, <i>available at</i> https://web.archive.org/web/19970614041044/http://home.netscape.com/eng/ssl3/ssl-toc.html and https://web.archive.org/web/19970617034012/http://home.netscape.com/eng/ssl3/3-SPEC.HTM#1 (“SSL-Standard”)
1022	U.S. Patent No. 8,756,673 (“Barrus”)
1023	Yuan et al., “Layered access control for MPEG-4 FGS video,” Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429), Barcelona, Spain, doi: 10.1109/ICIP.2003.1247012 (“Yuan”)
1024	U.S. Patent App. Pub. No. 2005/0060316 (“Kamath”)
1025	RESERVED
1026	<i>Apple Inc. v. Ericsson AB</i> , IPR2022-00618, Paper 2 (Feb. 25, 2022)
1027	<i>Apple Inc. v. Ericsson AB</i> , IPR2022-00618, Paper 9 (July 8, 2022)
1028	<i>Apple Inc. v. Ericsson AB</i> , IPR2022-00618, Paper 11 (Sept. 1, 2022)
1029	<i>Apple Inc. v. Ericsson AB</i> , IPR2022-00618, Paper 13 (Nov. 28, 2022)
1030	<i>Apple Inc. v. Ericsson AB</i> , IPR2022-00618, Paper 14 (Nov. 28, 2022)
1031	<i>Apple Inc. v. Ericsson AB</i> , IPR2022-00618, Paper 17 (Dec. 30, 2022)
1032	<i>In the Matter of Certain Electronic Devices, Including Mobile Phones, Tablets, Laptops, Components Thereof, and Products Containing the Same</i> , Inv. No. 337-TA-1376, Order No. 25 (Mar. 13, 2024)
1033	<i>In the Matter of Certain Electronic Devices, Including Mobile Phones, Tablets, Laptops, Components Thereof, and Products Containing the Same</i> , Inv. No. 337-TA-1376, Complaint, Exhibit 24 (October 12, 2023)

III. MY UNDERSTANDING OF PATENT LAW

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

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

24. I understand that, in this proceeding, Petitioners have the burden of proving that the Challenged Claims of the ’178 Patent are unpatentable by a preponderance of the evidence. I understand that “preponderance of the evidence” means that a fact or conclusion is more likely true than not true.

25. I understand that, in IPR proceedings, claim terms in a patent are given their ordinary and customary meaning as understood by a person of ordinary skill in the art (“POSITA”) in the context of the entire patent and the prosecution history pertaining to the patent. If the specification provides a special definition for a claim term that differs from the meaning the term would otherwise possess, the specification’s special definition controls. If a claim element is expressed as a “means” for performing a specified function, I understand that it covers the corresponding structure described in the specification and equivalents of the described structure. I have applied these standards in preparing the opinions in this declaration.

26. I understand that determining whether a particular patent or printed publication constitutes prior art to a challenged patent claim can require determining the effective filing date (also known as the priority date) to which the Challenged

Claim is entitled. I understand that for a patent claim to be entitled to the benefit of the filing date of an earlier application to which the patent claims priority, the earlier application must have described the claimed invention in sufficient detail to convey with reasonable clarity to the POSITA that the inventor had possession of the claimed invention as of the earlier application's filing date. I understand that a disclosure that merely renders the claimed invention obvious is not sufficient written description for the claim to be entitled to the benefit of the filing date of the application containing that disclosure.

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

A. Anticipation

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

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

B. Obviousness

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

31. I understand that a patent claim is obvious if the differences between the subject matter of the claim and the prior art are such that the subject matter as a whole would have been obvious to a person of ordinary skill in the relevant field at the time the invention was made. Specifically, I understand that the obviousness question involves a consideration of:

- the scope and content of the prior art;
- the differences between the prior art and the claims at issue;
- the knowledge of a person of ordinary skill in the pertinent art; and
- if present, objective factors indicative of non-obviousness, sometimes referred to as “secondary considerations.” I understand that before the International Trade Commission (“ITC”), in Investigation No. 337-TA-1376 (“1376 ITC Investigation”), the Patent Owner initially asserted that the inventions of the

'178 Patent were met with substantial commercial success and alluded to evidence of revenue associated with products that Patent Owner says are covered by the claims of the '178 Patent. I also understand that Patent Owner never identified any such evidence of revenue and provided no further explanation. To my knowledge, the Patent Owner has not asserted any other secondary considerations with respect to the '178 Patent. I understand that I may be asked to evaluate evidence related to secondary considerations if it is ever provided by Patent Owner and raised in this proceeding.

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

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

- combining prior-art elements according to known methods to yield predictable results;
- simple substitution of one known element for another to obtain predictable results;

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

34. I understand that for a single reference or a combination of references to render the claimed invention obvious, a POSITA must have been able to arrive at the claimed invention by altering or combining the applied references.

IV. BACKGROUND

A. '178 Patent

35. The '178 Patent (EX-1001) relates to “Web-based delivery of files over the public Internet, referred to as over-the-top (OTT) delivery,” and “encryption key rotation for live streaming media.” '178 Patent, 1:15-23. The '178 Patent describes a server that streams segments of encrypted media content to a client. '178 Patent, 1:38-42, 1:63-65. The '178 Patent explains that different segments in a media stream may be encrypted with different content encryption keys. '178 Patent, 1:45-65, 2:62-3:8. The '178 Patent describes the client communicating with a license server to pre-fetch content encryption keys ahead of key rotation boundaries in the media stream, and then applying a retrieved key to decrypt a corresponding segment of the media stream. '178 Patent, 11:3-25, 11:38-62. “The transitioning between use of different keys is also referred to” in the '178 Patent “as key ‘rotation.’” '178 Patent, 1:63-65.

B. Prosecution History of the '178 Patent

36. In May 2015, the Examiner identified claims 1-20 as allowable in a first office action. '178 FH (EX-1002), at pages 67-71. The office action cited only U.S. Patent Application Publication No. US 2005/0060316 to Kamath (EX-1024). In October 2015, the Examiner allowed the application, noting that “the prior art of record [Kamath] does not teach detecting key rotation boundaries and issuing

requests to a license server ahead of the key rotation boundaries.” EX-1002, at pages 29-31.

37. I am informed that in February 2022, Apple Inc. filed an *inter partes* review petition challenging all claims (claims 1-20) of the ’178 Patent in proceeding number IPR2022-00618. I reviewed documents filed in IPR2022-00618. The Board instituted *inter partes* review on grounds that used the same references used in Grounds 3 and 5, described in this declaration—namely, the Peterka, Bocharov, Peterka308, and Chen references. EX-1026, at pages 1-2; EX-1028, at page 32. The parties in IPR2022-00618 settled shortly after institution, and the proceeding was terminated before any Final Written Decision issued. EX-1031, at pages 2-3. I have been informed that Petitioners were not involved in IPR2022-00618.

V. PERSON OF ORDINARY SKILL IN THE ART (“POSITA”)

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

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

39. The '178 Patent involves technology in the field of information security systems that employ cryptographic keys to encrypt/decrypt digital data. I have been asked to provide my opinions as to the state of the art in this field by June 2011. I use this timeframe because the face of the '178 Patent indicates an earliest claimed priority date of June 23, 2011. Whenever I offer an opinion in this declaration about the knowledge of a POSITA, the manner in which a POSITA would have understood the claims of the '178 Patent or its description, the manner in which a POSITA would have understood the prior art, or what a POSITA would have been led to do based on the prior art, I am referencing the June 2011 timeframe, even if I do not say so specifically in each case.

40. I understand that the Patent Owner may attempt to prove that the alleged invention recited in the Challenged Claims was conceived at some time prior to the earliest claimed priority date on the face of the patent. At the time of this declaration, I am unaware of the Patent Owner having alleged any earlier conception date or produced any evidence to establish any earlier conception date.

41. In my opinion, a POSITA in the June 2011 timeframe would have had a bachelor's degree in computer science, computer engineering, electrical engineering, or a related field, and 2-3 years of practical engineering experience

relating to information security systems that employ cryptographic keys to encrypt/decrypt digital data. Additional education could substitute for professional experience, or significant experience in the field could substitute for formal education. The POSITA's level of skill would not have differed in a manner material to the grounds between the '178 Patent's earliest possible priority date (in 2011) and its actual filing date (in 2014).

42. By June 2011, I held a bachelor's degree and a master's degree in electrical engineering, and I had over thirty-five years of experience with high-technology systems related to military, commercial, and consumer communication systems and networks, including information security systems that employ cryptographic keys to encrypt/decrypt digital data. Therefore, I was a person of more than ordinary skill in the art during the relevant timeframe. However, I worked with many people who fit the characteristics of the POSITA, and I am familiar with their level of skill. When developing the opinions set forth in this declaration, I assumed the perspective of a POSITA, as set forth above.

VI. CLAIM INTERPRETATION

43. I understand that determining whether a claimed invention is novel and non-obvious requires comparing the prior art to the claim. In accordance with the standards for interpreting patent claims terms discussed in Section III above, I have interpreted the terms in the claims in accordance with their ordinary and customary

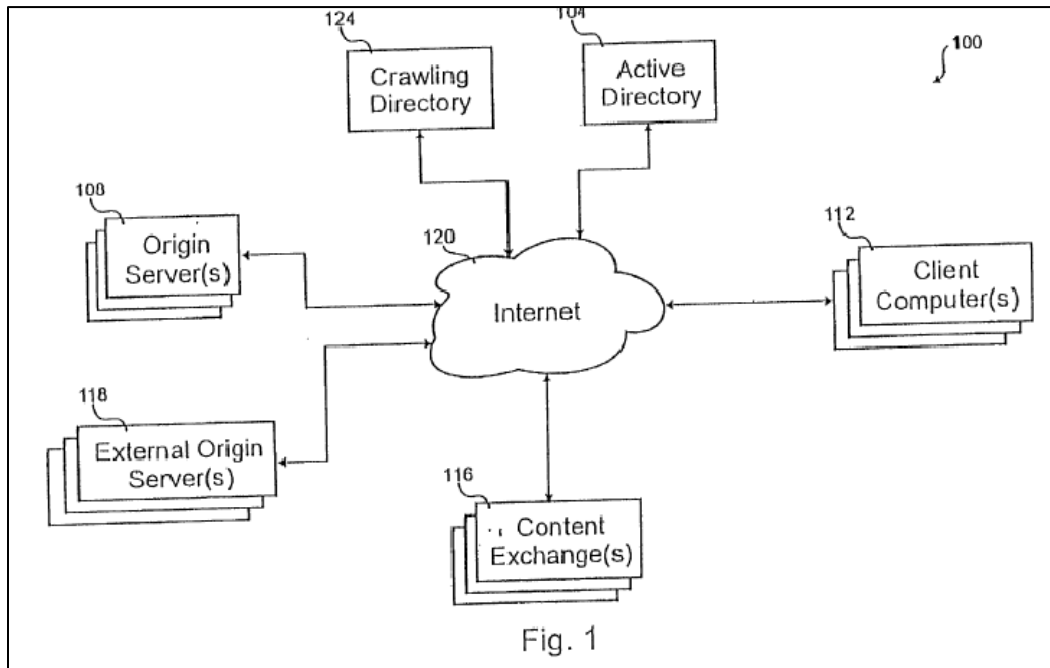
meaning as understood by a POSITA in the context of the '178 Patent (including the descriptions in the specification) and its prosecution history.

VII. PRIOR-ART REFERENCES

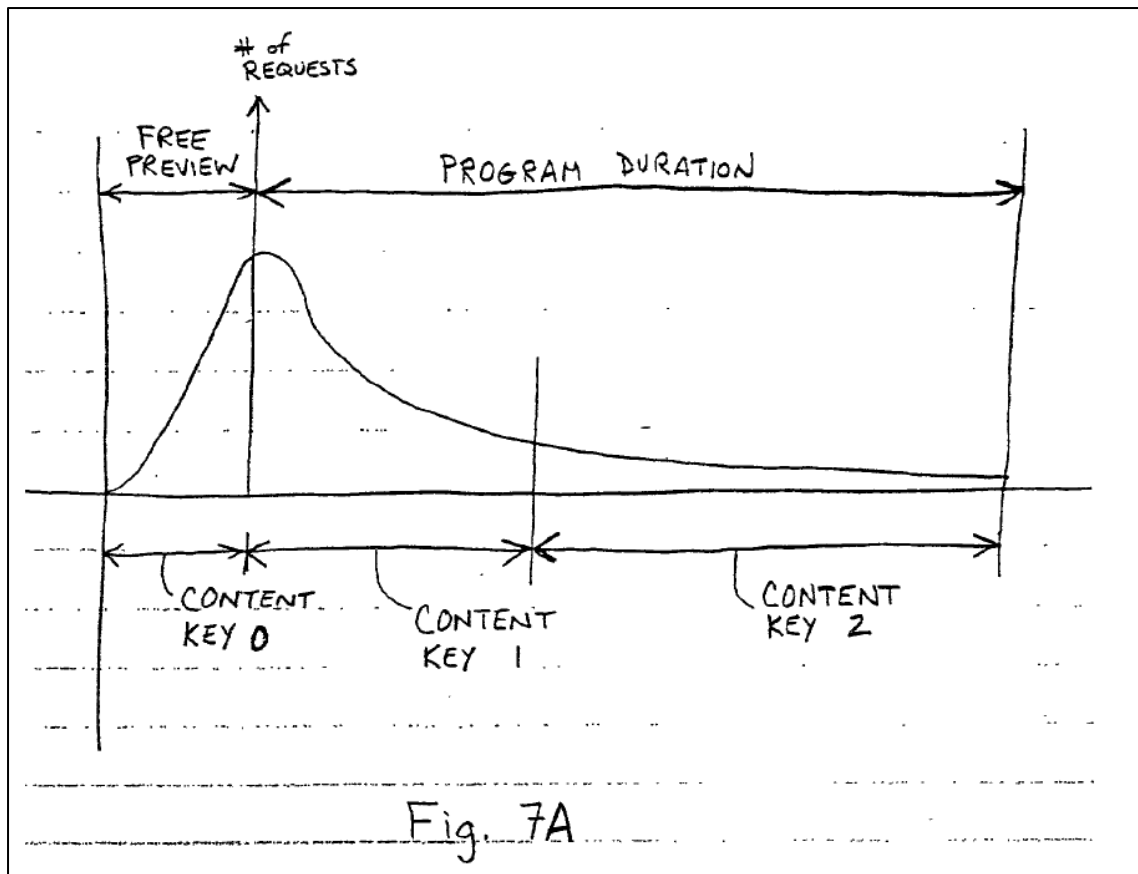
44. I briefly introduce in this Section below various prior-art references discussed in this declaration.

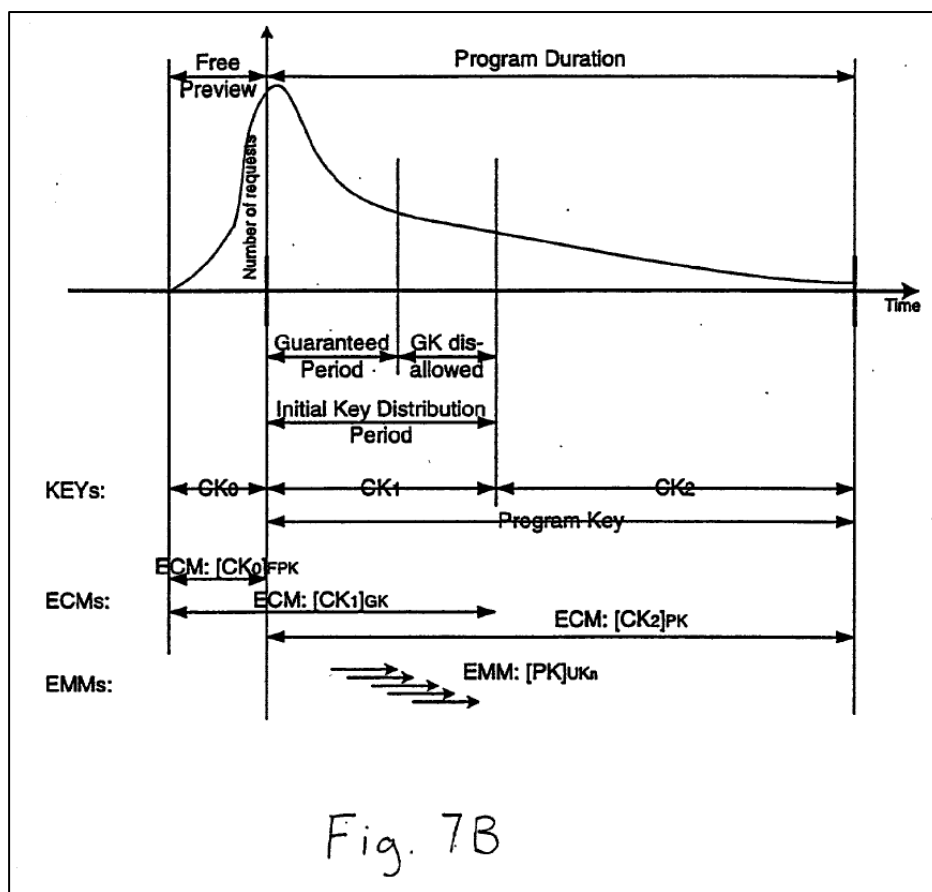
A. Peterka

45. Peterka (EX-1005) is directed to a multimedia distribution system for securely delivering streaming media to authorized devices over a network. Peterka, Abstract (“[V]arious distribution methods can be accomplished using encryption keys to distribute program content.”), FIG. 1, [0038] (“Referring to FIG. 1, a block diagram of a content distribution system 100 is shown.... A particular client computer 112 is shown interacting with the active directory 104 to select a content object for downloading. The object can be played during download if it is a streaming media or can be stored for later display. The content object could be various types of information, such as audio, video, or data that is available to be downloaded from the network. Furthermore, it can be used for multicasting and/or unicasting.”), [0006]-[0012] (summarizing “a method of multicasting program material is provided by providing encrypted program material for distribution to a client”). FIG. 1 of Peterka is reproduced below:



46. Peterka teaches its “client” devices downloading streaming media from a content distribution system comprising servers. Peterka, [0038]. Peterka also teaches dividing content into “segments” that are each encrypted with a different key prior to streaming. Peterka, [0080]-[0082] (describing a “server” that “may divide the content into pay segments and assign segment keys to them”: “All consumer computers would negotiate a key for each segment in order to keep track of how many segments they watched.”), [0099]-[0101] (describing one way of dividing a program into “purchasable segments”), [0091]-[0093] (describing FIG. 7A, which shows a program that is divided and encrypted with a series of “content key[s]”), [0122]-[0125] (describing FIG. 7B, which also shows a program that is divided and encrypted with a series of “content key[s]”), FIGs. 7A-7B. Figures 7A and 7B of Peterka are reproduced below:





47. Peterka discloses servers distributing content to clients using various purchase mechanisms, including “pay-by-time (PBT)” —in which users pay for the time duration of content that is actually watched. Peterka, [0053] (“Pay-By-Time (PBT)—A purchase mechanism in which the client pays for the time duration of the content that was actually watched. The discrete time increments may have different durations for different programs or services.... Some sporting and music events are of this type.”). Users may pay for, *e.g.*, each media segment requested. Peterka, [0080] (“All consumer computers would negotiate a key for each segment in order to keep track of how many segments they watched.”), [0099] (describing one way

of implementing acquisition of keys for a program divided into “purchasable segments”).

48. Peterka teaches streaming encrypted media content segments to clients in packets, *e.g.*, Real-Time Transport Protocol (“RTP”) packets. Peterka, [0043]. Peterka teaches that the media content may be, for example, in the well-known MPEG-4 format. Peterka, [0119]. Peterka also teaches that the media content its system distributes may be “live content.” Peterka, [0058] (“Multicast—Multicast is similar to the existing TV broadcast. It delivers the same content to one or more consumers at the same time. This is usually scheduled and can be live content.”).

49. Peterka also teaches using content packets to signal key changes to clients. Peterka, [0116]-[0120] (teaching various ways to provide “key change” notifications, including, for example, in RTP packet headers). Note that Peterka’s disclosed “signaling methods” inform a client precisely when to start using a new key to decrypt received content, *regardless of how that key was obtained by the client*. Peterka’s *separate* teachings regarding obtaining keys (independent of Peterka’s signaling teachings) include several ways for a client to obtain a next key before it is needed for decryption. Peterka, [0105]-[0115]. For example, Peterka describes a “Push” model for key distribution (where a server keeps track of keys at each client and pushes keys as needed), a “Pull” model for key distribution (where a client keeps track of its keys and actively requests keys as needed), and a

combination “Push-Pull” model for key distribution (*e.g.*, where a server pushes keys to only those clients which have confirmed receipt of a previous key). Peterka, [0105] (describing the different models), [0106]-[0110] (pull model), [0111]-[0112] (push model), [0113]-[0115] (push-pull model). The POSITA would have understood that all of these methods for obtaining keys were compatible with Peterka’s signaling methods for informing clients *when* to change keys used for decryption. See my additional discussion below in paragraphs 115-117. For the purpose of this declaration, I focus my analysis on Peterka’s pull-model for key distribution—in which a client keeps track of its own keys and sends requests to a server to obtain keys as needed.

50. For each encrypted content segment, Peterka teaches a client obtaining the key used to encrypt the segment—*e.g.*, a content key (“CK”)—and applying that key to decrypt the segment before playing it back. Peterka, [0011] (“One embodiment allows providing a key and utilizing the key to accomplish the encrypting of a remaining portion of the program. In addition, one embodiment may allow the user to purchase the program content as well as provide the user with a key that is operable to decrypt the encrypted portion of the program.”), [0091]-[0093] (describing an implementation of FIG. 7A where clients obtain “[c]ontent keys” that “are required to decrypt” various portions of “program content.”), [0127] (explaining that “clients... utilize the content key to decrypt the encrypted program content”),

FIG. 7A. For embodiments where keys themselves are encrypted, Peterka teaches issuing (1) an encrypted version of a key that was used to encrypt a segment of media content and (2) at least one secondary key used to encrypt the key. Peterka, [0094]-[0104]. An authorized user's client applies the secondary key(s) to decrypt the encrypted key to obtain the content key, which is then used to decrypt content. *Id.*

51. Peterka teaches that each key, along with information about the key, is distributed to a client device in an Entitlement Control Message (“ECM”) or an Entitlement Management Message (“EMM”). Peterka, [0104], [0121]-[0125], FIG. 7B. While Peterka teaches generally that EMMs and ECMs perform different types of messaging, Peterka also discloses they can perform similar messaging. *See, e.g.*, Peterka, [0104] (“These [EMM] group-addressed multicast messages play the role of Entitlement Control Message (ECM) messages”), [0125] (“CK2 may be encrypted with the UK [(unique key)] and delivered in the form of an EMM directly to the client instead of the ECM form shown in the figure.”).

52. As discussed, in Peterka's “pull model” for obtaining keys, “each client keeps track of the keys and their expiration times and actively requests new keys before the current keys expire so as to avoid service interruptions.” Peterka, [0105]-[0106], FIG. 8. To mitigate excessive server loads at media segment boundaries in such an embodiment—that is, an embodiment (as Peterka teaches) where clients actively request and negotiate keys for segments—the POSITA would have

understood Peterka to teach distributing *multiple* keys (*e.g.*, multiple ECMs) to a client (*e.g.*, for the current *and* next media segment) in response to such a request. Peterka, [0080] (when discussing what Peterka calls “a quasi-PBT approach” where “a caching server may divide the content into pay segments and assign segment keys to them”: “All consumer computers would negotiate a key for each segment in order to keep track of how many segments they watched. This generates a large load on the caching server on the segment boundaries. This can be mitigated with a key management approach where each rabbit is given keys to the current as well as the next segment. This will give the caching server enough time to distribute the next segment keys during the current segment.”);¹ *see also* Peterka, [0130] (discussing FIG. 7B: “Any user that purchases the program content during the guaranteed period will receive the next necessary decryption key within the initial key distribution.”), FIG. 7B.

53. Peterka also discloses embodiments in which clients can adjust the quality (*e.g.*, bit rate) of the media its users view (for example, for an MPEG-4

¹ The use of the term “rabbit” in paragraph [0080] is a typographical error. Based on Peterka’s disclosures and the context that Peterka is describing in the surrounding text, a POSITA would have understood Peterka to be referring to “clients” receiving “keys to the current as well as the next segment.” *See* Peterka, [0078]-[0080].

format stream, per Peterka, [0119]) but does not specify the mechanism by which clients receive information that facilitates such adjustments—such as information regarding what quality encodings are available to stream. Peterka, [0054] (“Pay-By-Quality (PBQ)—Content may be offered at a different quality (i.e., bit rates), either as separate streams or as layered streams with each additional layer adding quality to the content. The client may report the highest bit rate it can consume and be offered to purchase the content at that quality or less. The server may also adjust the rate in real time based on the immediate state of the network.... The client may also adjust the bit rate perceived by the user, in accordance with user selection.”).

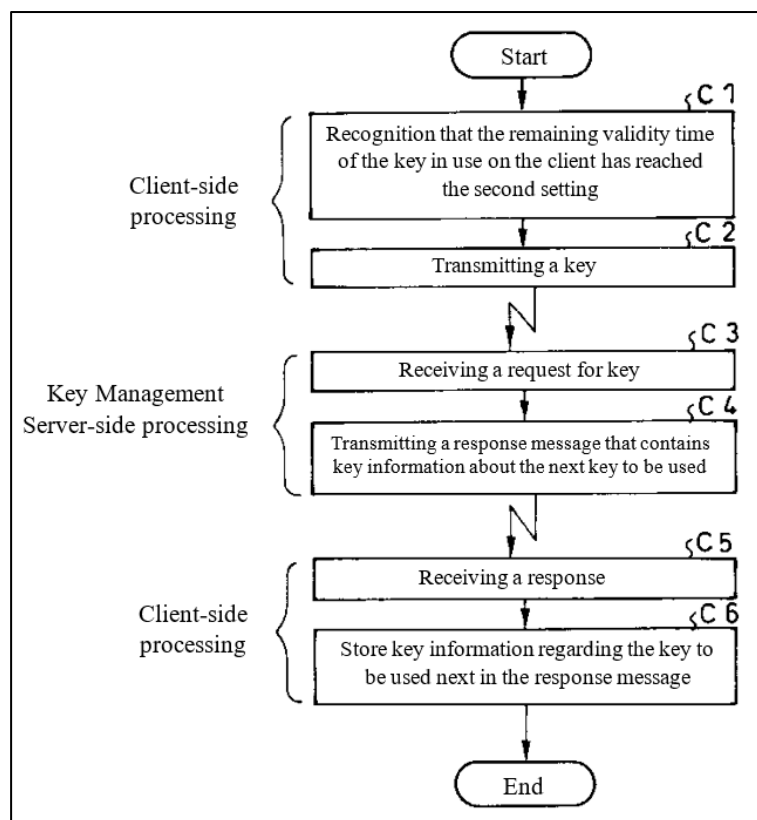
54. Based on Peterka’s disclosures and a POSITA’s background knowledge, a POSITA would have understood Peterka’s teachings about content being available at different network rates or quality levels (Peterka, [0054])—and users having the option to choose between various offered media qualities—would be implemented with Peterka’s various disclosed purchase mechanisms, including pay-by-time (Peterka, [0078]-[0082])—for example in an implementation where users are charged based on the time duration of content viewed, *at a chosen quality/bit rate*.

B. Suzuki

55. Suzuki (EX-1006) relates to providing media securely using a multicast packet delivery system that encrypts data with keys that are “periodically” changed

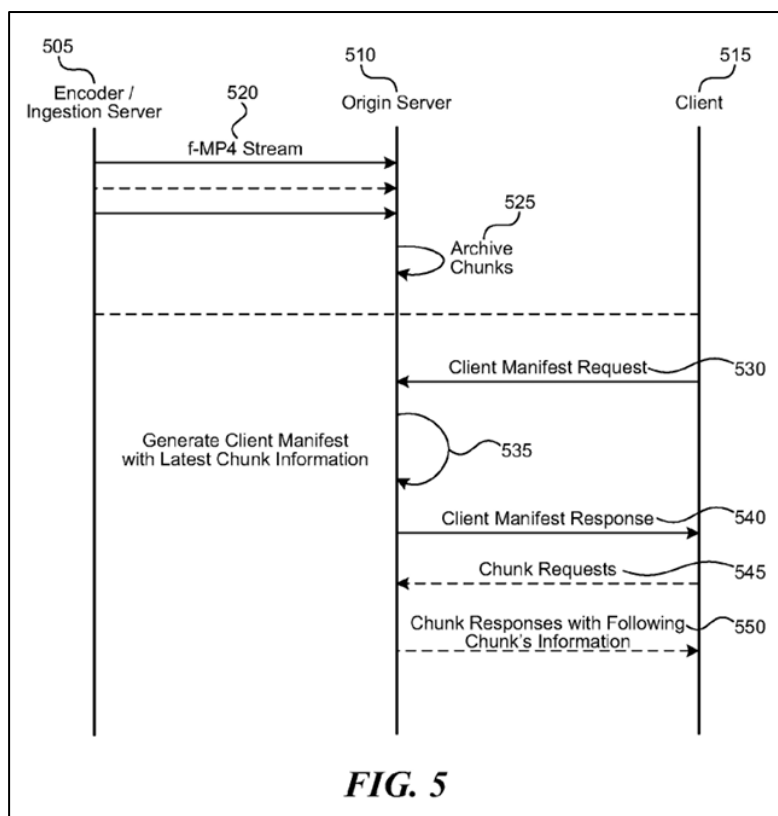
at “regular intervals.” Suzuki, Abstract, [0001] (“The present invention relates to a multicast distribution system that prevents eavesdropping by encrypting multicast distribution using a different key at regular intervals, and to a key exchange method in a multicast distribution system for controlling and managing the exchange of keys (encryption key/decryption key).”), [0004]-[0005]. Suzuki teaches “enabl[ing] real-time data processing” of encrypted packets “by eliminating the *confirmation delay*” during key changes. Suzuki, [0011]-[0012]. A POSITA would have understood Suzuki’s system that enables real-time data processing to be well-suited for streaming, *e.g.*, video. Suzuki, [0144] (“[U]nauthorized persons can be prevented from continuing to watch” content “by changing the key.”).

56. Suzuki addresses “confirmation delay” using a client “key management means... that transmits a key information request to the key management server when the remaining validity time of the key in use reaches” a particular value, and a key management server “key management means” that “returns a response message” with “the decryption key to be used next, its key number, and its remaining validity time to the client.” Suzuki, [0016]. In Suzuki, the client performs “processing” that includes “recogni[zing]” that the value of “the remaining validity time has decreased and reached a [particular] setting value.” Suzuki, [0036], [0059], FIG. 4. Figure 4 of Suzuki is reproduced below:



C. Bocharov

57. Bocharov (EX-1007) is directed to the streaming of media in fragments from a server to a client, in which “manifest” files containing metadata information about each fragment (*e.g.*, encoding information about the fragment) are embedded in fragments along with “media content”—thus “eliminat[ing] the usage of a typical control channel.” Bocharov, Abstract, [0001], [0007], [0014], [0030]-[0031], [0053], FIGs. 3-5. Figure 5, reproduced below, is a flow diagram depicting interactions between origin server 510, encoder 505, and client 515 in Bocharov’s system:



58. As shown, “encoder [server] 505 continuously provides media data 520 to... origin server 510.” Bocharov, [0053]. “The media data may include fragments of an MP4 stream based on a live event.” *Id.* “The origin server 510 receives a manifest request 530 from a client 515,” and “generates 535 a client manifest based on the latest media fragment information.” *Id.* “The origin server 510 provides a client manifest response 540 to the client 515,” “[t]he client 515 then sends one or more media fragment requests 545, and the origin server 510 responds 550 with the requested media fragment and potentially information about subsequent media fragments.” *Id.*; *see also* Bocharov, [0006] (“[L]ive media often cannot be cached using existing protocols, and each client requests the media from the same server or

set of servers.”), [0014]-[0016] (“Each fragment contains metadata information that describes the encodings available on the server and the encoding of the fragment in addition to the media content of the fragment for playback by the client.”), [0021]-[0031] (“The client interface component 150 receives client requests for media fragments and provides manifest data and media fragments to clients.”), [0049]-[0057] (“[T]he system builds a manifest to satisfy the client request based on the fragments received and the encoder information received when the system initially requested encoder manifests.”), [0060] (“[I]nformation may be included in the manifest provided to the client.”).

59. After sending a client the initial client manifest response, Bocharov teaches servers “includ[ing]” additional manifests to accompany the media content in fragments the client receives. Bocharov, [0023]-[0024] (“When a client initially connects to the system 100, the client may send a request for a client manifest.... The build client manifest component 160 builds a manifest to satisfy a client request that includes information about each of the encodings available from the system 100 and fragments stored by the system up to the current time. The build client manifest component 160 also provides a manifest to include with each media fragment that provides information to the client about the current media fragment as well as potentially subsequent fragments. By combining the initially received manifest with subsequent manifests provided with each media fragment, the client can build an up

to date manifest that includes complete information about the media event from the start up until the current time.”), [0007] (“Each fragment contains metadata information that describes the encodings available on the server and the encoding of the fragment in addition to the media content of the fragment for playback by the client. The server may provide fragments in multiple encodings so that the client can, for example, switch quickly to fragments of a different bit rate or playback speed based on network conditions.”), [0014]-[0016] (“Each fragment contains metadata and media content. The metadata may describe useful information about the media content, such as the bit rate at which the media content was encoded, where the media content fits into a larger media element (e.g., this fragment represents offset 1:10 in a 10-minute video clip), the codec used to encode the media content, and so forth. The client uses this information to place the fragment into a storyboard of the larger media element and to properly decode and playback the media content.”).

60. Bocharov teaches its “client manifest” including information useful for the client to request media fragments from a server for either pre-recorded or live programming. For example, Bocharov teaches providing timestamps for each fragment archived up to the point of the request, which the client uses to compose a URL for requesting the fragment from the server. Bocharov, [0030]-[0031] (“The

client manifest lists the decoding information as well as information for all the fragments that the server has archived so far.”).

61. Bocharov’s client manifest further identifies available encodings of the media content. Bocharov, [0049] (“Based on the manifest, the client can begin requesting fragments in whichever encoding the client chooses.”). Thus, for programs encoded in multiple bit rates, the client references Bocharov’s manifest to “begin requesting fragments in whichever encoding the client chooses.” *Id.* Bocharov’s manifests enable switching between different encodings mid-stream. “For example, the client may initially select a low bit rate encoding and select higher bit rate encodings for subsequent fragments until network bandwidth limits the client’s ability to receive the fragments at a bit rate.” *Id.*; *see also* Bocharov, [0023]-[0025], [0030]-[0031], [0049], [0052], [0057], [0060].

62. Additionally, Bocharov’s manifests facilitate a “payment model” where clients pay by quality and where “switching to particular bit rates” is “enable[d] or disable[d]” “based on whether the conditions of the payment model have been met.” Bocharov, [0060].

D. Peterka308

63. Peterka308 (EX-1008) describes techniques for decrypting secured video segments for playback from a DVR. Peterka308, Abstract, FIGs. 2-6. In Peterka308, a DVR 104 “receives encrypted content from [a] content source 102.”

Peterka308, [0021]. “[C]ontent protection module 106” decrypts the content so that the DVR 104 “may re-encrypt the content and record the re-encrypted content by storing it on a hard drive 202.” *Id.* “DRM component 204” of the DVR can be configured to “generate[] a unique CEK [(content encryption key)] for each segment of copy never content at a predefined time interval during recording.” Peterka308, [0024].

64. Peterka308’s CEK need not be “stored in the hard drive 202 or other media storage.” Peterka308, [0025]. “Rather, a time stamp, which indicates the relative time value from the beginning of the recording to the start of the segment, is generated and stored at the time that each unique CEK is determined.” *Id.* “Each time stamp is dynamically added to [a] content license as the recording progresses.” *Id.* “During playback, the master key and the time stamp for a particular segment may be utilized, at least in part, to derive the CEK for that segment so that the encrypted content for that segment stored in the hard drive 202 or other media storage may be decrypted.” *Id.*; *see also* Peterka308, [0026]-[0028].

E. Chen

65. Chen (EX-1009) describes techniques for encrypting video on-demand (“VOD”) “material with a changing cryptographic key.” Chen, [0008]; *see generally* Chen, [0008]-[0011], [0027]-[0028], [0034]-[0038], [0035], FIGs. 2A-2D. Chen’s “technique for implementing rapid cryptographic key changes uses any

of four different basic schemes... for breaking up (subdividing, segmenting, sectioning) the content to be encoded/encrypted into a plurality of ‘crypto periods’, covered by different cryptographic keys”: “fixed crypto period,” “fixed number of packets,” “fixed ‘marker’ count,” and “random crypto period.” Chen, [0034]; *see also* Chen, [0037]-[0047].

VIII. THE CHALLENGED CLAIMS ARE UNPATENTABLE IN LIGHT OF THE PRIOR ART IDENTIFIED IN THE PETITION

66. I have been asked to provide my opinions concerning whether the Challenged Claims of the ’178 Patent are anticipated or would have been obvious to a POSITA in light of the prior-art references identified in the Petition. For the reasons explained below, it is my opinion that each of the Challenged Claims would have been obvious to a POSITA.

67. A table summarizing the Grounds demonstrating unpatentability, including the primary references and claims in each Ground, is provided below. The primary references I rely upon are introduced above in Section VII.

Ground and Reference(s)		Claims
1	Peterka (EX-1005)	1-6, 10, 12, 16-18, 20
2	Peterka, Suzuki (EX-1006)	1-6, 10, 12, 16-18, 20
3	Peterka, Bocharov (EX-1007)	1-6, 10, 12, 14, 16-18, 20
4	Peterka, Suzuki, Bocharov	1-6, 10, 12, 14, 16-18, 20
5	Peterka, Bocharov, Peterka308 (EX-1008), Chen (EX-1009)	7-8, 19

68. I have been informed that the '178 Patent claims priority to a provisional application filed June 23, 2011. I understand that Petitioners treat this date as the earliest effective filing date of each Challenged Claim for the purposes of the analysis in the petition. As noted above in paragraph 39, my opinions in this declaration are also based on this June 2011 timeframe.

69. I also understand that each primary prior art reference that I applied in this declaration in the Grounds described below qualifies as prior art to each Challenged Claim even if Patent Owner could establish that it was entitled to the priority date of June 23, 2011, based on the filing and publication dates noted below:

Reference	Filed	Published
Peterka	10/26/2001	11/21/2002
Suzuki		06/10/2004
Bocharov	03/16/2009	09/16/2010
Peterka308	08/22/2007	10/30/2008
Chen		5/12/2004

70. Additionally, based on my review of the file history, I note that none of the above references were cited or applied in a rejection during the original prosecution of the '178 Patent.

A. Ground 1: Peterka Renders Claims 1-6, 10, 12, 16-18, and 20 Obvious

71. As I discussed above in § VII.A, Peterka discloses a multimedia distribution system that uses a “Pay-By-Time (PBT)” mechanism for viewing

content (in which “the client pays for the time duration of the content that was actually watched”) and a “pull method” for key distribution (where clients monitor key expiration times to request new keys before they expire). Peterka, [0053], [0078]-[0082], [0105]-[0106], [0110]. As shown below, Peterka discloses or renders obvious every element of independent claims 1 and 16.

72. While Peterka does not use the term “manifest files,” retrieving “manifest” files from a content delivery server (*e.g.*, a server that also delivers media content) was a well-known, conventional way to provide clients access to information about the media (metadata) that facilitated streaming—*e.g.*, information about encoding and media quality options. See my analysis below for element [1C], in § VIII.A.1.a.iv, discussing corroborating evidence for this proposition, including Bocharov (EX-1007), [0024], [0030]-[0031]; Lewis (EX-1011), 1:28-29, 7:6-9, 7:38-45; Pestoni (EX-1012), 8:39-9:19; Sood (EX-1013), Abstract, [0019]-[0020], [0046]-[0047]; Chen050 (EX-1014), 15:31-36, 15:66-16:8; Stockhammer (EX-1015), 134-135; Van Lancker (EX-1016), Abstract, §§ 1, 3; 3GPP-AHS (EX-1017), § 12.2.5; and Pantos (EX-1018), pages 4-5, 9.

73. Thus, as discussed in § VIII.A.1.a.iv, Peterka—which discloses servers that “adjust the rate in real time based on the immediate state of the network” and clients that are able to “adjust the bit rate” of streamed media delivered to the user (Peterka, [0054]), *e.g.*, in the MPEG-4 format (Peterka, [0119])—would have been

understood to teach to a POSITA the retrieval of conventional manifest files from content delivery servers to enable such adjustments.

1. Claim-by-Claim Analysis

a. Claim 1

i. [1PRE] “A method for handling secure distribution of content comprising:”

74. I am informed that Patent Owner has agreed before the International Trade Commission (“ITC”) that Claim 1’s preamble is not limiting. EX-1032, pages 15-17. Regardless, Peterka discloses a method for handling secure distribution of content, for example live video content. Peterka, [0012] (“a method of providing program material to multiple clients”), [0083] (“In block 304[,], encrypted material is provided for distribution to a client[,],... [f]or example, ...by a content exchange such as a metropolitan video exchange or an origin server.”), [0094] (“Under various distribution methods... encryption keys can be distributed to clients to facilitate reception of the program content.”), [0058] (“can be live content”), FIGs. 1, 3.

ii. [1A] “initiating a media playback request and receiving a playback request response;”

75. In Peterka, clients request access to media content—*e.g.*, under Peterka’s pay-by-time distribution scheme—and then receive encryption keys to gain access to at least a portion of a media stream. Peterka, [0080]-[0082] (“When the client negotiates encryption keys with the caching server, it will start receiving” segments of selected “content” from an ongoing show/event); *see also* Peterka,

[0091]-[0092] (“FIG. 7A illustrates a plot showing the number of *requests*^{2/} for a program” and “the distribution of content keys”), [0141]-[0143] (disclosing embodiments where “*request* [for program content is] sent by the client” and the client receives encryption keys and encrypted content), FIGs. 7A-7B, 20A-20B.

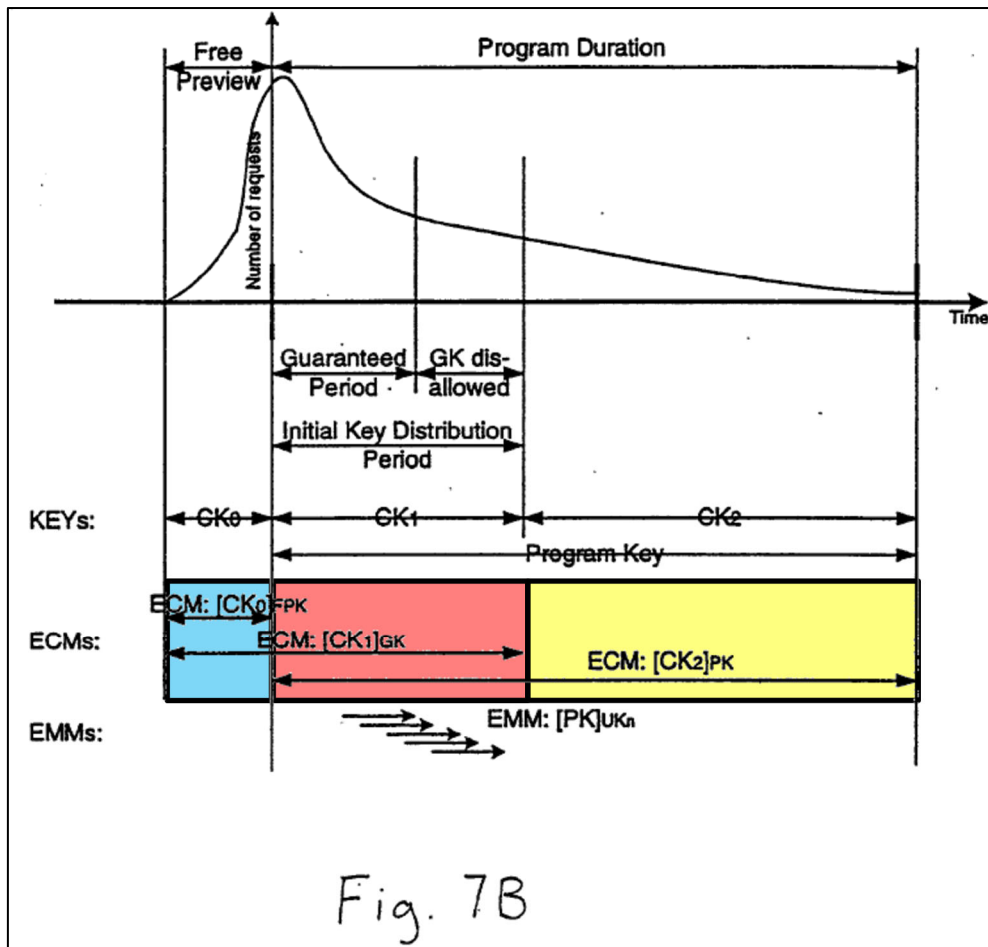
76. Peterka’s disclosed client-initiated requests to access media, *e.g.*, program content, are each “*initiating a media playback request*” as recited in claim 1. This interpretation is consistent with how “playback requests” are described in the ’178 specification, which generally uses the term to refer to requests to playback media, *without requiring* any particular registration or authentication mechanisms for the client. ’178 Patent, 2:15-21 (“A client device handles the secure distribution of content by a process including initiating a media playback request and receiving a playback request response...”), 9:49-52 (“In step 302, the client 110 issues a playback request to the WFM 102. In one embodiment, the initial playback request may require registration of the client 110, user login, or some other authentication mechanism.”).

77. As discussed above in § VII.A, the POSITA would have understood Peterka to teach distributing multiple keys (*e.g.*, current and next) to a client when it negotiates for access to a media content segment. Peterka, [0080] (clients are “given keys to the current as well as the next segment”), [0130] (describing FIG. 7B, which

² Emphasis added unless otherwise indicated.

shows clients receiving a “content key which is utilized to decrypt encrypted program content” for a current period as well as “the next necessary decryption key within the initial key distribution”), FIG. 7B. In particular, Peterka describes subdividing a media stream into “segments” and applying a different content encryption key to each segment. Peterka, [0080]-[0082], [0099]-[0101], [0091]-[0093], [0122]-[0125], FIGs. 7A-7B. Peterka also describes “a key management approach where each [client] is given keys to the current as well as the next segment.” Peterka, [0080]. Peterka teaches that such an approach “can... mitigate[]” “large load[s]” on a server that occur at “the segment boundaries” of a media stream—including in a distribution model where users pay for discrete time increments and a server is likely to receive key requests from many clients at each media segment boundary. *Id.*; *see also id.*, [0053] (“Pay-By-Time (PBT)—A purchase mechanism in which the client pays for the time duration of the content that was actually watched.”).

78. Peterka FIG. 7B (annotated below)—“showing exemplary distribution of cryptographic keys during portions of a program” (Peterka, [0021])—would have confirmed to the POSITA that this same key management approach was applicable to any distribution scheme in which key requests from many clients are expected at segment boundaries.



79. As shown above, Figure 7B shows two content keys CK_0 and CK_1 distributed in separate ECMs³ for each request received during a free preview period (blue), and two content keys CK_1 and CK_2 distributed in separate ECMs for each request received during an initial key distribution period (red). ECMs with CK_1 are

³ Peterka's illustration of one content key contained in each ECM is consistent with Peterka's teachings regarding the fields in each ECM or EMM, which are repeatedly characterized as containing information about a single key. Peterka, [0121] (e.g., "the key in the message").

distributed to clients *before* the period in which CK₁ is used (red). Peterka, [0130] (describing user “receiv[ing] the next necessary decryption key within the initial key distribution”). ECMs with CK₂ are distributed to clients *before* the period in which CK₂ is used (yellow). *Id.* Providing a client with a content key for the next segment in response to a client’s request to begin playback during the current segment as taught in Peterka was desirable (and indeed obvious), since this would predictably reduce the volume of requests that the server needed to respond to ahead of the upcoming segment boundary and help reduce disruptions in content distribution due to latency caused by any network congestion that arises from many clients using the network to transmit requests. Additionally, the POSITA would have understood it to be beneficial for the client to have both the current and the next content keys for scenarios in which servers become temporarily unable to distribute the content key for the next segment, which would disrupt the client’s ability to continue decrypting after the next segment boundary.

80. Peterka teaches that each key (along with information about the key, as I discuss below in the context of Element [1B]) is “distributed to the client in response to the request by the client” in an ECM or EMM. Peterka, [0104]-[0106] (describing key distribution implementations using ECMs and EMMs), FIG. 8 (showing steps confirming Peterka’s teaching of providing keys to clients in response to client requests: “receive request for a cryptographic key from a client”

(810) and “distribute the key to the client in response to the request” (818)), [0121] (explaining that “[i]n multicasting program content such as audio and visual material, entitlement management messages and entitlement control messages can be sent from a server to client computer,” and teaching fields of, *e.g.*, key metadata that are present in each ECM or EMM), [0122]-[0125] (describing FIG. 7B and teaching various ways to deliver keys in an ECM or EMM), FIG. 7B (showing three content keys—CK₀, CK₁, and CK₂—each distributed in three respective ECMs).

81. A POSITA would have understood that the ECMs or EMMs that are sent to a client in response to a particular “*media playback request*” collectively constitute a *playback request response* received by the client, as claimed. To the extent that the term “playback request response” is interpreted to require data transmitted in a single message—a narrow interpretation that assumes (without justification) that the claim term “response” is interchangeable with the word “message”—it would have been obvious to the POSITA in view of Peterka’s teachings to package multiple entitlement/management messages (ECMs or EMMs) into a single “combined” multicast message, thus delivering multiple keys. Peterka, [0104] (describing “for improved efficiency,” “multiple EMMs... combined into a single multicast message”), [0121] (describing “multiple EMMs... delivered in a single multicast message”). Peterka’s disclosures confirm that such packaging of entitlement/management messages (ECMs or EMMs) for “improved efficiency”

was a viable option with which the POSITA would have reasonably expected success. *Id.* This is further confirmed by a POSITA’s knowledge that it was conventional to transmit multiple keys (*e.g.*, current and next) in a single message in response to a client request for keys. For documentary evidence corroborating this knowledge of a POSITA, see Suzuki, [0076]-[0077], FIGs. 5-6.

iii. **[1B] “parsing content information from the playback request response, the content information including content encryption keys, content encryption key identifiers, and content encryption key expiration times;”**

82. The ’178 Patent uses the term “parsing” when describing general retrieval of information from a playback request response. *See* ’178 Patent, 2:15-21, 9:52-56. No specific manner of parsing is described. The claim also does not require any specific manner of parsing.

83. As discussed above at § VIII.A.1.a.ii, Peterka teaches a server that returns to a client device ECMs or EMMs that are sent in response (collectively a *playback request response*) to a request to access or play at least a portion of a stream (*media playback request*). The ECMs or EMMs contain information responsive to the request described above in § VIII.A.1.a.ii. Based on Peterka’s teachings—including Peterka’s teachings that the keys and key metadata (*e.g.*, expiration times) are kept track of and used for decrypting content (Peterka, [0105]-[0106])—a POSITA would have recognized that the client *parses* the ECMs or

EMMs to distinguish among data included in the response. Peterka, [0104] (describing keys transmitted in ECMs or EMMs), [0121] (describing fields of metadata in each ECM or EMM), [0122]-[0125] (describing FIG. 7B, which illustrates content keys transmitted in ECMs, and alternative implementations of keys transmitted in ECMs or EMMs), FIG. 7B.

84. Peterka describes, for example, that each ECM or EMM includes separate fields for “a sequence number or a time stamp to protect against replay attacks,” “the type of key... such as a content key, a group key, a program segment key, a service key, etc.,” “the type of key used to encrypt the key in the message,” “the time remaining in the lifetime of the key,” and “a key parity bit matching [a] parity bit in the RTP packet” containing encrypted content. Peterka, [0121].

(1) Content Information

85. Peterka’s ECMs or EMMs that are sent in response (collectively a *playback request response*) to a request to access or play at least a portion of a stream (*media playback request*) provide each type of “*content information*” recited in Element [1B]:

a) Content Encryption Keys

86. The ’178 specification describes “*content encryption keys*” as keys that are used “for encrypting a content item” or “decrypting the content item.” ’178 Patent, 1:45-58 (“A disclosed method includes generating a series of content

encryption keys and providing them serially to a packaging server *for encrypting a content item*, wherein each content encryption key is provided upon expiration of a period of use of a serially preceding content encryption key. The packaging server generates packaged content for delivery to client devices via a content delivery network, the packaged content including or accompanied by key expiration information usable by the client devices to identify transitions between sections of the packaged *content encrypted by different ones of the content encryption keys*. The method further includes providing the content encryption keys to a license server for delivery to the client devices *for use in decrypting the content item.*”), 3:51-52 (“The client uses the content encryption keys to decrypt the associated content.”). Claims 1 and 16 make clear that “content encryption keys” must be used at least for decrypting content.

87. In Peterka, ECMs or EMMs carry keys to client devices, including “content key[s]” (“CKs”) that were “used to encrypt the content itself” and are used by the client to decrypt content (*content encryption keys*). Peterka, [0101]; *see also* Peterka, [0106] (indicating that “the program content is distributed for decryption by the client utilizing the cryptographic key that was requested”), [0116] (indicating that “the content key is utilized for decrypting the program content throughout the course of the distribution of the program content”), [0121] (indicating that a “key” is “included in the message, such as a content key”). As discussed above in

§ VIII.A.1.a.ii, multiple keys are conveyed to the client in response to a playback request, and as discussed above these are parsed.

88. Elements [1D]-[1F], which I discuss further in §§ VIII.A.1.a.v-VIII.A.1.a.vii below, require content encryption keys to be “used” or “applied” for decrypting content. As I noted above in § VII.A, Peterka teaches embodiments where content keys are themselves encrypted and must be decrypted before use. Peterka, [0094]-[0104]. In the 1376 ITC Investigation, I understand that Patent Owner has argued that the term “content encryption keys” is met by encrypted keys even though such encrypted keys must themselves first be decrypted before they can be used or applied for decrypting content. Under Patent Owner’s interpretation, Peterka’s CKs are “content encryption keys” even when they are encrypted.

89. Alternatively, the POSITA would have understood that another well-known, conventional option was to transmit Peterka’s content keys to clients in clear (or unencrypted) form. This would satisfy the term “content encryption keys” under a narrower interpretation which excludes *encrypted* keys. For corroboration of the POSITA’s knowledge regarding the conventional option to send keys that are not encrypted, see, for example the Suzuki reference, which describes sending clients keys for decrypting content without encrypting them. *See* Suzuki, [0016], [0072], FIG. 5.

b) Content Encryption Key Identifiers

90. The '178 specification says each content encryption key may be assigned a “unique” “*content encryption key identifier*.” '178 Patent, 3:29-32. As discussed below, Peterka meets this limitation even if the term “content encryption key identifier” is construed to mean an identifier that *uniquely* identifies a key.

91. Peterka teaches that each ECM or EMM includes “a *sequence number* or a *time stamp* to protect against replay attacks.” Peterka, [0121]. Because the sequence number or time stamp is unique to each ECM or EMM (in order to prevent replay attacks, as Peterka states), and each ECM or EMM carries a different content encryption key (as I discussed above in § VIII.A.1.a.ii), the sequence number or time stamp each constitutes a unique *content encryption key identifier* because it can be used to uniquely identify the key that was in the ECM or EMM.

92. This interpretation of content encryption key identifier is consistent with the '178 specification's teaching of “content encryption key identifiers” that are “selected based on a segment number of the prepared content” (and thus apply to an entire media segment). '178 Patent, 3:34-35, 4:8-10; *see also* '178 Patent, 6:1-3 (“In another embodiment, the content encryption key identifiers are based off of segment numbers, as produced by the packager 104 during segmentation.”); 6:17-19 (same). A POSITA would have recognized that one way to “select” an identifier based on a segment number of content (as the '178 Patent describes) was by using

the segment number of the content *itself* as an identifier for the key associated with that segment. That would have been a simple, straightforward option for selecting an identifier for a key—one that required minimum processing resources—and one that a POSITA would have reasonably expected success with, because a POSITA understood that the key was used to encrypt that one segment (and not another).

93. As discussed below in § VIII.A.1.a.iii(2), the received keys' respective identifiers are conveyed in response to a playback request, and as discussed above they are parsed.

94. Moreover, a POSITA would have been motivated to implement Peterka to receive and use a key identifier to identify the content key parsed from each ECM or EMM. In systems where multiple keys were managed and utilized—like Peterka's (Peterka, [0080], [0122], [0130], FIG. 7B)—a POSITA would have (1) recognized the need to use key identifiers to differentiate between keys and facilitate the use of the correct key for the correct segment, and (2) been motivated to implement Peterka to use such key identifiers. Indeed, FIG. 7B itself shows the delivery (in ECMs) of a series of content keys (CKs) identified by what a POSITA would understand were content key identifiers (CK₀, CK₁, and CK₂), and would have motivated a POSITA to implement Peterka's received keys to have associated key identifiers. Peterka, [0122], FIG. 7B.

95. A POSITA would have reasonably expected success implementing Peterka's keys to have key identifiers that could be used to facilitate keeping track of the keys and applying the correct key to a given segment, because this technique was conventional and well-known at the time of the alleged invention. For evidence corroborating the well-known use of received key identifiers to help keep track of keys in a multi-key environment, see, for example, the Suzuki reference, which discloses a key exchange method in a multicast content distribution system. *E.g.*, Suzuki, [0013] (disclosing a client using "key information management table" with "key number" for each key), [0016] (same, and disclosing a client receiving a "key number" for each key), [0030]-[0031] (disclosing use of a "key information management table" at a client device, with a "key number for identifying the key"), [0043]-[0048] (disclosing a client using a "key number" to distinguish between keys and apply the correct key to decrypt content), [0080]-[0084] (same), FIG. 5.

96. For additional corroborating evidence, see the Pestoni reference, which discloses embodiments of a content protection system. *E.g.*, Pestoni, 12:16-35 ("In various embodiments, playback device 100 may be configured to decrypt each segment 440*a-c* with the respective decryption key of key box 420. In some embodiments, key box 420 may specify a key identifier for each key within the key box. Likewise, each encrypted segment 440*a-c* may include metadata (e.g., header information) that specifies the identifier of the decryption key that is to be utilized

to decrypt that segment. For a given segment that is to be decrypted, playback device 100 may be configured to determine that the key identifier specified by that segment's metadata matches a key identifier of a particular decryption key within key box 420. In response to that determination, playback device 100 may be configured to utilize that particular decryption key to decrypt the segment. In various embodiments, key box 420 may include an array of key identifier-decryption key pairs. In these embodiments, the playback device may be configured to compare the key identifier specified by a segment's metadata to the array of key identifier—decryption key pairs to identify the decryption key that is to be used to decrypt that content item.”).

c) Content Encryption Key Expiration Times

97. Peterka discloses that an ECM/EMM includes a field indicating “the time remaining in the lifetime of the key” contained in the ECM/EMM, which constitutes a *content encryption key expiration time*. Peterka, [0121]; *see also* Peterka, [0106] (“[T]he key and *its expiration time* are distributed to the client...”), [0105] (“Under the pull model, each client keeps track of the keys and *their expiration times* and actively requests new keys before the current keys expire...”), FIG. 8.

98. This interpretation of “content encryption key expiration times” is consistent with the ’178 specification’s discussion of requesting new content

encryption keys before the end of the current key's lifespan—just like in Peterka. '178 Patent, 6:66-7:2 (“If the client 110 has the right to view the content, the license server 106 provides the content encryption key, content encryption key lifespan (or expiration), and the location of the encrypted content.”), 9:47-62 (discussing the client: “FIG. 3 is a flow chart 300 describing a process for detecting the rotation of content encryption keys and retrieving updated content encryption keys.... In step 304, the client 110 retrieves and parses the content information, including content formats, content bitrates, content locations, content encryption keys, and content encryption key expiration.”), 11:7-25 (discussing the client: “If the current content encryption key has not yet expired, processing proceeds to step 314. If the current content encryption key needs to be expired, processing proceeds to step 312 where the key is ‘rotated’, i.e., the current content encryption key is expired and a new one is put into use. The client 110 first checks whether the content encryption key corresponding to a key change notification in step 308 has already been retrieved by prefetching as described below with reference to step 318. If the new content encryption key has already been retrieved, the current content encryption key is replaced with that pending content encryption key. If the new content encryption key has not been retrieved yet (e.g., due to early expiration of the current content encryption key for security reasons, or content encryption key retrieval failures in step 318), the client 110 must issue a new content encryption key request to the

license server 106 to retrieve the new content encryption key, then replace the current content encryption key with the newly retrieved content encryption key.”); *see also* ’178 Patent, 6:25-32 (discussing the content packaging engine: “In one embodiment, before the content encryption key expires, the packager 104 requests a new key from the WFM 102. In one embodiment, the new content encryption key has the same lifespan as the previous content encryption key. The new content encryption key is made available by the WFM 102 to the packager 104 before the previous content encryption key has expired to allow for uninterrupted encryption.”), 8:58-61 (discussing the content packaging engine: “In step 210, executed after step 208 or after step 206 when no new content encryption key is received, the packager 104 checks the expiration of current content key to see if a new key should be applied.”), 9:20-25 (discussing the content packaging engine: “In step 216, the packager 104 checks to see if the current content encryption key is going to expire in the near future. In one embodiment, when the current content encryption key is about to expire, the packager 104 proactively requests a new content encryption key ahead of time to prevent any interruption of encryption services.”).

99. As discussed below in § VIII.A.1.a.iii(2), the received keys’ respective expiration times are conveyed in response to a playback request, and as discussed above they are parsed.

**(2) Peterka Teaches a Playback Request Response
Comprising “Content Information” for
Multiple Segments**

100. As discussed, each ECM or EMM in Peterka is parsed for the required content information in Element [1B]. Peterka discloses sending multiple ECMs or multiple EMMs in response (*playback request response*) to a given client-initiated request to access media (*media playback request*). See my discussion above in § VIII.A.1.a.ii. A POSITA would have understood that in the *playback request response*, each ECM or EMM would contain one content key and its corresponding *identifiers* and *expiration time* described above at §§ VIII.A.1.a.iii(1) [subsection b)]-[subsection c)]. Peterka, [0121]. Thus, Peterka satisfies the requirement that each “*playback request response*” includes multiple “*content encryption keys*,” “*content encryption key identifiers*,” and “*content encryption key expiration times*,” as recited in Element [1B].

101. I understand based on Patent Owner’s positions in the 1376 Investigation, Patent Owner may argue that because no *single* ECM or no *single* EMM in Peterka includes multiple “content encryption keys,” “content encryption key identifiers,” and “content encryption key expiration times,” Peterka does not meet Element [1B]. In my view, such an argument improperly focuses on how the parts of the multiple-key response—*i.e.*, the entitlement messages (ECMs and EMMs) in Peterka—are named. As noted before, the claims recite a “response”—

not a “message.” Such an argument also ignores that Peterka teaches sending more than one key (and thus more than one ECM or more than one EMM) in response to a given client-initiated request to access media, and further ignores that these two or more ECMs or EMMs are collectively the “playback request response.” Further, as discussed above in paragraph 81, it would have been obvious to POSITAs to package multiple ECMs/EMMs in a single multicast message that includes multiple “content encryption keys,” “content encryption key identifiers,” and “content encryption key expiration times,” based on Peterka’s express teachings and a POSITA’s background knowledge. Peterka, [0104], [0121]. For evidence corroborating a POSITA’s background knowledge, see Suzuki, [0076]-[0077], FIGs. 5-6.

iv. [1C] “retrieving content and manifest files from a content delivery server;”

102. Peterka discloses a client device (*e.g.*, client computer 112) *retrieving content from a content delivery server* (*e.g.*, origin 108, a content exchange 116, or caching server). Peterka, [0038] (“A particular client computer 112 is shown interacting with the active directory 104 to select a content object for downloading.”), [0039] (“[T]he origin servers 108 determine the preferred source to direct the client computers in order to download content objects.... That source can be origin processor 108 itself or one of the content exchanges 116.”), [0040] (“Content objects of an external origin processor 118 can be preloaded to a content exchange(s) allocated to provide those content objects. To decrease latency when a

content object is requested for the first time, the active directory 104 can call the external origin processor 118 to determine the content objects available from the external origin server 118.”), [0060] (“Origin Content (OCS)—Server at a content provider’s computer that provides content.”), [0071] (OCS “becomes part of the distribution network”), [0083] (“In block 304 encrypted material is provided for distribution to a client. For example, such encrypted material could be provided by a content exchange such as a metropolitan video exchange or an origin server from which the content originated.”), [0135] (“Referring to FIG. 19 a system can be seen for implementing rules and conditions for providing content in a multicasting environment. System 1900 shows a client server network comprised of at least one client 1908 coupled to a server such as origin content processor 1904 via a network 1916 such as the internet. In addition, FIG. 19 shows a caching processor 1912 and an authorization center 1920 also coupled to the network. The origin content server is intended to illustrate a server which stores or controls access to program content. For example, such content could be multimedia or it could be a movie for distribution via a webcasting system. The caching processor 1912 can be utilized in this multicasting environment to store a copy of the program content that originated on the origin content server.”), FIGs. 1, 19. For additional context, see my discussion above in § VII.A.

103. Peterka does not explicitly use the term “manifest files.” However, manifest files were well-known to the POSITA as conventional features of media streaming platforms, such as the ones taught in Peterka, which stream, *e.g.*, segments of MPEG-4 format media at various bit rates. Peterka, [0053]-[0054], [0119].

104. For corroboration, see, for example, Lewis, which explains that “[c]onventionally, many streaming platforms rely on a manifest file retrieved as a static playlist file from a web host.” Lewis, 1:28-29 (“Conventionally, many streaming platforms rely on a manifest file retrieved as a static playlist file from a web host.”), 2:64-3:4 (describing a conventional media streaming implementation to which Lewis’s manifest teachings can be applied: “For example, the live video footage may be encoded into MPEG transport stream (TS) fragment files of a fixed length, such as 10 seconds, for ease of distribution using streaming protocols such as HTTP streaming, IP multicast, P2P streaming. As new live footage is captured and approved, new video segments may be encoded and stored within live video segments 175, which may then be distributed over content delivery network 135.”), 7:6-9 (describing a known use for manifest files: “Additionally, multiple bit-rate streams may be encoded and referenced in the generated manifest files to allow graceful degradation to lower bit-rate video in response to adverse network conditions.”), 7:38-45 (describing known use of manifest file to “implement enhanced security features, such as limiting or granting access or providing keys

based on time windows or rental periods, HTTP cookie or login status, client device identification, or other criteria. For example, if the enhanced security checks fail, then the client may be redirected to a video showing how the user can remedy any failed security checks.”).

105. Consistent with the above descriptions in Lewis which corroborate a POSITA’s background knowledge regarding manifest files, manifest files provided, *e.g.*, “information about the encodings available from” a media streaming system “and fragments stored by the system up to the current time,” and other useful information that clients could access and use to help facilitate streaming, as further confirmed by the Bocharov (EX-1007) reference. Bocharov, [0023]-[0024] (“The build client manifest component 160 builds a manifest to satisfy a client request that includes information about each of the encodings available from the system 100 and fragments stored by the system up to the current time.”), [0030]-[0031] (describing a “typical client manifest” that includes “decoding information as well as information for all the fragments that the server has archived”).

106. Numerous other references corroborate a POSITA’s background knowledge of manifest files and their uses in media streaming platforms.

- Pestoni, 8:39-9:19 (“[F]or each encrypted content item that is to be acquired by the playback device, there may be a respective *manifest* that specifies the internal content and/or structure of that encrypted content item. Such

manifests are illustrated as manifests 220*a-n*.... [T]he manifests may be identified by updateable playlist 215.... [E]ach of the manifests may be obtained prior to the time at which the corresponding encrypted content is to be played. For instance, the playback device 100 may obtain manifest 220*a* before encrypted content item 240*a* is scheduled to be consumed (e.g., according to updateable playlist 215). Manifest 220*a* may specify the license (e.g., via a license identifier) that is required to decrypt encrypted content items 240*a*.”)

- Sood (EX-1013), Abstract (“The client requests uniform chunks of media from the server that include a portion of the media.... Each chunk contains metadata information that describes the encoding of the chunk and media content for playback by the client. The server may provide chunks in multiple encodings so that the client can switch quickly to chunks of a different bit rate or playback speed.”), [0019]-[0020] (“The manifest assembly component 130 builds a manifest that describes the media element to which received media content belongs.... During streaming, particularly live content, a server cannot provide a complete manifest because the event is still ongoing. Thus, the server provides as much of the manifest as it can through the metadata in the media chunks.... The client-side assembled manifest allows the client to play the media event back as on-demand content as soon as the event is over,

and to skip around within the media event as it is going on.”), [0046]-[0047] (“[T]he adaptive streaming system determines which encodings are available based on a subscription or other payment model. For example, a content provider may charge more for a high definition (HD) version of a live event than a standard definition (SD) version... the client may enable or disable switching to particular bit rates based on whether the conditions of the payment model have been met.... The adaptive streaming system may request and receive media content in a variety of encodings. In some embodiments, the adaptive streaming system uses custom MP4 boxes. The Motion Picture Experts Group (MPEG) version 4 standard provides for boxes within the format that can contain custom data. The MP4 extension is the file format commonly associated with this version of content. The system may leverage boxes to include the custom metadata and media content chunks. Other media formats provide similar customization of content within a container and may be used.”)

- Chen050 (EX-1014), 15:31-36 (“In addition, encapsulation unit 30 may form a manifest file, such as a media presentation descriptor (MPD) that describes characteristics of the representations. Encapsulation unit 30 may format the MPD according to extensible markup language (XML).”), 15:66-16:8 (“Manifest file 66 may include data indicative of the subsets of representations

68 corresponding to particular representation groups, as well as common characteristics for the representation groups. Manifest file 66 may also include data representative of individual characteristics, such as bitrates, for individual representations of representation groups. In this manner, a representation group may provide for simplified network bandwidth adaptation. Representations in a representation group may be indicated using child elements of a representation group element of manifest file 66.”).

- Stockhammer (EX-1015), 134 (stating, in a paper that I understand was published in February 2011⁴: “Figure 2 shows a possible media distribution architecture for HTTP-based streaming. The media preparation process typically generates segments that contain different encoded versions of one or several of the media components of the media content. The segments are then hosted on one or several media origin servers typically, along with the media presentation description (MPD). The media origin server is preferably an HTTP server such that any communication with the server is HTTP-based (indicated by a bold line in the picture). Based on this MPD metadata information that describes the relation of the segments and how they form a

⁴ http://crossmark.crossref.org/dialog/?doi=10.1145%2F1943552.1943572&domain=pdf&date_stamp=2011-02-23.

media presentation, clients request the segments using HTTP GET or partial GET methods. The client fully controls the streaming session, i.e., it manages the on-time request and smooth playout of the sequence of segments, potentially adjusting bitrates or other attributes, for example to react to changes of the device state or the user preferences.”), 135 (“explaining that “MPEG’s draft DASH solution is heavily based on 3GPP’s AHS.”).

- Van Lancker (EX-1016), Abstract (stating, in a paper that I understand was presented in July 2011⁵: “MPEG started with the standardization of an HTTP streaming protocol, defining the structure and semantics of a manifest file and additional restrictions and extensions for container formats.”), §§ 1 (“Recently, a new media delivery technique, called HTTP adaptive streaming, was introduced showing an interesting combination of the features of real-time streaming protocols and HTTP progressive download. Various proprietary implementations are already available: Microsoft’s Smooth Streaming, Apple’s HTTP Live streaming, and Adobe’s Dynamic HTTP Streaming. Almost all current proprietary solutions for HTTP streaming define the structure and semantics of a manifest file, describing the high-level structure of the media content in terms of representations and temporal segments.... Currently, MPEG is standardizing HTTP adaptive streaming as

⁵ <https://ieeexplore.ieee.org/document/6012149>.

media delivery protocol, Dynamic Adaptive Streaming over HTTP (DASH), which is based on 3GPP Adaptive HTTP Streaming.”), 3 (“As discussed in Sect. 1, HTTP streaming solutions make use of a manifest file. It may describe different representations (e.g., different bit rates, languages, or resolutions) of the same media content. Typically, these representations correspond to different media resources or track combinations within one media resource.... Examples of manifest files using Media Fragment URIs to point to representations are shown in Listings 1 and 2, using the Media Presentation Description (MPD) and M3U8 syntax respectively.”).

- 3GPP-AHS (EX-1017), § 12.2.5 (describing attributes and elements of “Media Presentation Description (MPD)” manifest in 3GPP adaptive HTTP streaming standard from June 2010).
- Pantos (EX-1018), 4-5 (describing M3U playlist file for HTTP Live Streaming in March 2011).

107. Additionally, the ’178 specification itself lists conventional manifest file formats that were well-known at the time of the alleged invention—“m3u8,” “Smooth Streaming,” and “DASH MPD.” ’178 Patent, 4:16-20, 7:56-59, 10:57-60. Several of the references discussed in the previous paragraph corroborate that the file formats listed by the patent were described in the prior art. *E.g.*, Stockhammer,

pages 133-135; Van Lancker, Abstract, §§ 1, 3; 3GPP-AHS, § 12.2.5; Pantos, pages 4-5.

108. As further support for the claimed manifest files corresponding conventional manifest files commonly used by media streaming platforms, in the 1376 ITC Investigation, I understand that Patent Owner has argued that Element [1C] encompasses conventional “manifest files”—*e.g.*, a “MPD” file that describes “different resolutions and bandwidth for each media file.” *See* EX-1033, 13-14.

109. Peterka renders obvious clients retrieving conventional manifest files from content delivery servers. Peterka discloses that media (*e.g.*, in MPEG-4 format) may be available at different “quality (i.e., bit rates)” and describes the “server... adjust[ing] the rate in real time based on the immediate state of the network,” and the “client... adjust[ing] the bit rate... in accordance with user selection.” Peterka, [0054], [0119] (“MPEG 4”); *see also* Peterka, [0059] (describing a “free preview” that “may be offered at a lower quality.”). Based on Peterka’s disclosures regarding network server and/or client-initiated bit-rate adjustments, it would have been obvious to implement Peterka’s clients to retrieve from the content delivery server both the content itself and manifest files describing the content, which at minimum included information regarding available “quality (i.e., bit rates)” for streaming media that would allow the disclosed bit-rate adjustments. Moreover, such an implementation would have been obvious in view

of a POSITA's knowledge (described above, and corroborated by the references discussing manifests, discussed above) because it was conventional for content servers to (1) create manifest files with information regarding how a particular media segment is encoded (*e.g.*, bit rates in the context of multi bit-rate streaming services) and (2) send such information to clients. *See, e.g.*, Bocharov, [0024], [0030]-[0031]; Lewis, 1:28-29, Pestoni, 8:39-9:19; Sood, Abstract, [0019]-[0020], [0046]-[0047]; Chen050, 15:31-36, 15:66-16:8; Stockhammer, 134-135; Van Lancker, Abstract, §§ 1, 3; 3GPP-AHS, § 12.2.5; Pantos, 4-5. Accordingly, a POSITA would have reasonably expected success implementing Peterka to include such functionality.

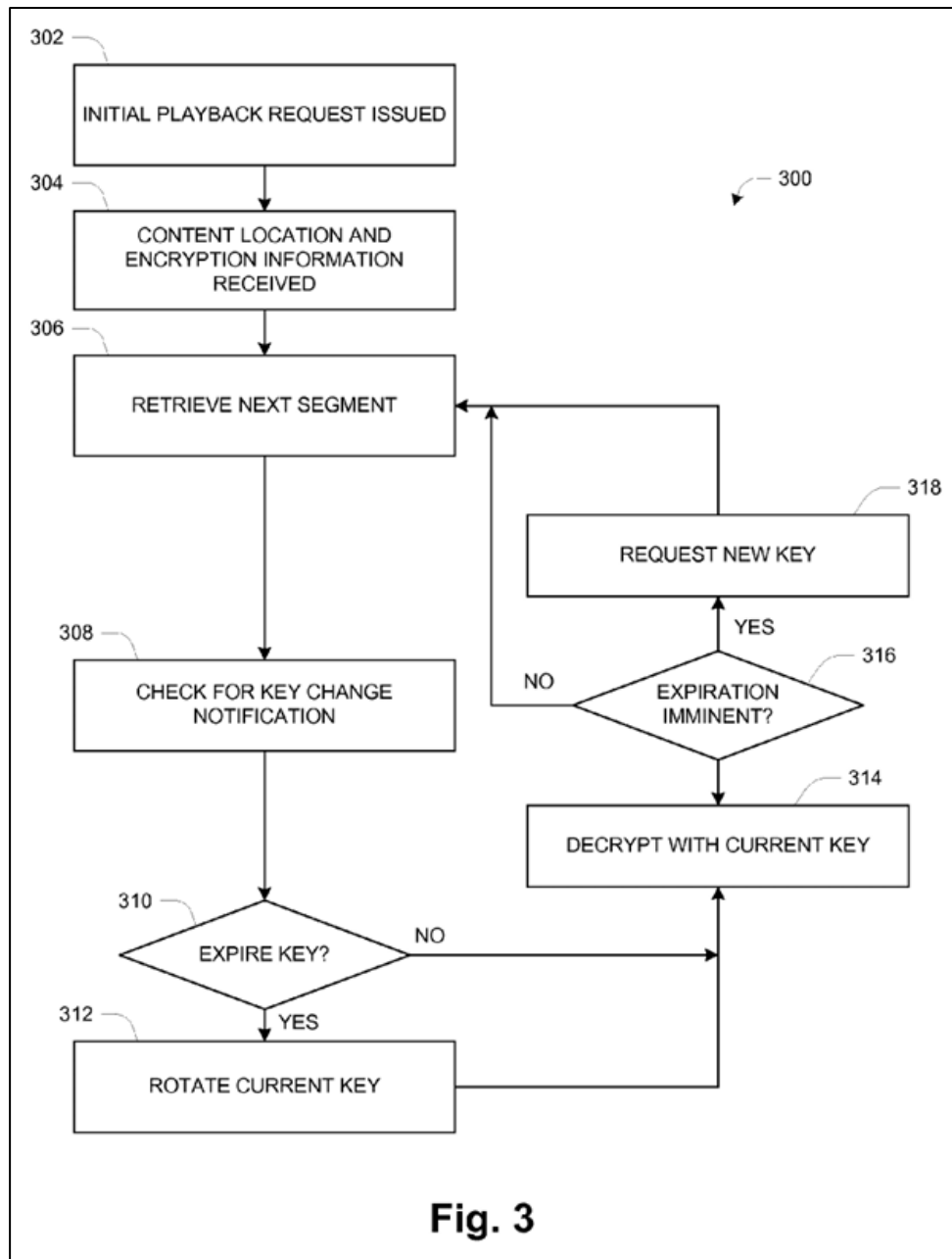
110. It also would have been obvious to the POSITA to implement Peterka's client, which retrieves media streams that were divided into segments (Peterka, [0080]-[0082], [0099]-[0101], [0091]-[0093], [0122]-[0125], FIGs. 7A-7B), to retrieve *multiple* manifest files from the content server, with each subsequent manifest file describing subsequent corresponding media segments. As discussed in, *e.g.*, the Bocharov reference, it was conventional for servers to send clients updated manifests with updated information regarding media segments that may not have been available in an initial manifest acquired when the client initially accessed a media stream. *See* Bocharov, [0024] ("The build client manifest component 160 also provides a manifest to include with each media fragment that provides information to the client about the current media fragment as well as potentially

subsequent fragments. By combining the initially received manifest with subsequent manifests provided with each media fragment, the client can build an up-to-date manifest that includes complete information about the media event from the startup until the current time.”), [0032]; *see also* Bocharov, [0023]-[0025]. At minimum, such an implementation—in which a separate manifest file describing information regarding a segment of media content was sent along with the segment itself—would have been obvious to reduce the complexity of parsing manifest files at a client device and having to associate various data with multiple media segments. A POSITA would have reasonably expected success implementing Peterka to include such functionality. Peterka thus renders obvious “*retrieving... manifest files from a content delivery server*,” as claimed.

v. [1D] “**detecting content encryption key rotation boundaries between periods of use of different content encryption keys in decrypting retrieved content;**”

111. The ’178 specification indicates that, for a series of media segments that are encrypted by successive encryption keys, “key rotation boundaries” occur when a new key is required to decrypt a segment regardless of whether the key change is due to normal period-based key expiration or due to some other reason. *See* ’178 Patent, 7:29-33 (“The lifespan of the new content encryption key may be aligned to the periodic use period boundaries of the previous content encryption keys. In one embodiment, the expiration of the new content encryption key is set to

the expiration time of the previous content encryption key.”). The ’178 specification describes detecting such key rotation boundaries by: (1) checking the expiration time of the current key (normal period-based expiration) or (2) checking for an explicit key change notification/signal. ’178 Patent, 9:47-49 (“FIG. 3 is a *flow chart 300 describing a process for detecting the rotation of content encryption keys* and retrieving updated content encryption keys.”). FIG 3 of the ’178 Patent—which includes multiple steps (including steps 308, 310, and 316)—is reproduced below:



See also '178 Patent, 10:39-42 (describing step 308 in FIG. 3: "Once the new segment is retrieved, processing proceeds to step 308 where the client 110 checks to see if a key change is signaled, i.e., a key change was requested outside of any fixed duration lifespan."), 11:2-7 (describing step 310 in FIG. 3: "In step 310, the client 110 *checks the expiration of current content key to see if a new key should be*

applied. There are two cases—*normal period-based expiration, and an explicit key change notification received in step 308 that applies to the current segment being decrypted.*”), 11:38-45 (describing step 316 in FIG. 3: “In step 316, the client 110 checks to see if the current content encryption key is going to expire in the near future or if a content encryption key change request for a future segment was detected in step 308. In one embodiment, when the current content encryption key is about to expire, the client 110 proactively requests a new content encryption key ahead of time to prevent any latency in or interruption of decryption services.”). No specific required manner of “detecting content encryption key rotation boundaries between periods of use of different content encryption keys in decrypting retrieved content” is described in the specification. The claim language also does not require any specific manner of “detecting.” The POSITA would have understood Element [1D] to cover either type of detecting described above—(1) checking the expiration time of the current key (normal period-based expiration) or (2) checking for an explicit key change notification/signal.

112. Peterka teaches detecting key rotation boundaries as claimed. As discussed above in § VII.A, Peterka discloses a pull model for key distribution, in which “each client keeps track of the keys and their expiration times and actively requests new keys before the current keys expire so as to avoid service interruptions.” Peterka, [0105]. Peterka teaches that “the server need not monitor

the status of the client's keys; rather, the responsibility of *determining when a new key is required* is "passed to the client." Peterka, [0106]. Peterka's clients "can *monitor*," for example, "the expiration time of the program segment key and send an individual key update request before the key expires." Peterka, [0114]; *see also* Peterka, [0080]-[0082] ("The client will *monitor* the expiration time for each pay period."). Peterka thus discloses client devices configured to *detect* upcoming *content encryption key rotation boundaries between periods of use of different content encryption keys*, based at least on monitored key expiration times that allow the client to request new keys ahead of the upcoming key rotation boundary.

113. Moreover, these detected boundaries are boundaries that occur *between periods of use of different content encryption keys*, as claimed, because the different content encryption keys are used to encrypt different media segments—*see* Peterka, [0080]-[0082], [0091]-[0093], [0099]-[0101], [0122]-[0125], FIGs. 7A-7B—in the same way that the '178 Patent specification describes. '178 Patent, 2:22-30 ("During ongoing retrieval of content, the client device detects content encryption key rotation boundaries *between periods of use of different content encryption keys in decrypting retrieved content*, issues requests to the license server ahead of a key rotation boundary to retrieve a second content encryption key to be used after a content encryption key rotation boundary is reached, and applies the second key for content decryption after the key rotation boundary is reached."), 1:45-56 ("A

disclosed method includes generating a series of content encryption keys and providing them serially to a packaging server for encrypting a content item, wherein each content encryption key is provided upon expiration *of a period of use of a serially preceding content encryption key*. The packaging server generates packaged content for delivery to client devices via a content delivery network, the packaged content including or accompanied by key expiration information *usable by the client devices to identify transitions between sections of the packaged content encrypted by different ones of the content encryption keys.*”).

114. Based on my review of filings in the IPR2022-00618 proceeding, I understand that Patent Owner previously argued that Peterka does not meet Element [1D] because, according to Patent Owner, Element 1[D] is not “directed to” detecting “normal period-based expiration” of keys. Patent Owner’s IPR2022-00618 Preliminary Response (marked EX-1027 in this proceeding), at pages 12-13 (“Limitation 1.4 is directed to the latter scenario, *i.e.*, detecting an explicit key change notification that applies to the current segment being decrypted, and that creates a key rotation boundary between normal periods of use, *i.e.*, before normal period-based expiration.”), 18 (Peterka’s “paragraph 116 does not teach that the client detects a key rotation boundary ahead of the normal expiration period that was already set for the current key, as limitation 1.4 requires.”). I understand that the Board in the IPR2022-00618 proceeding rejected this interpretation, and found that

Peterka’s pull-model-clients—which “keep[] track of... keys and their expiration times”—meet Element [1D]. *See* IPR2022-00618 Institution Decision (marked EX-1028 in this proceeding), at pages 20-23. In my opinion, the Board in IPR2022-00618 was correct to reject a narrow interpretation of Element [1D] that tries to exclude Peterka’s time-based boundary detection. Such a narrow interpretation of “detecting” is not in my opinion supported by the ’178 specification.

115. Even if a narrow interpretation were adopted here that excludes the time-based boundary detection disclosed in the ’178 Patent, Peterka also separately meets Element 1[D] because Peterka teaches boundary detection through use of “signaling method[s]” in the media stream that allow a client to precisely determine when to “begin *utilizing* [a] new content key” (Peterka, [0116]-[0117])—for example, an as-yet unused content key that was requested and received using Peterka’s pull model. In particular, Peterka teaches using “a predetermined bit” in the content stream “to indicate if an old or current content key should be used as opposed to a new content key which has recently been distributed to the client.” Peterka, [0117]. The POSITA would have understood that Peterka’s disclosure of a client “*check[ing]*” this “predetermined bit in a packet” to “determine the proper content key to use” *additionally meets* Element [1D]’s “detect[ion],” because the bit signals to the client a key change and signals to the client when to begin utilizing a new content key. Peterka, [0116]-[0121], FIG. 14. The POSITA would have also

understood that Peterka's teachings regarding "signaling" *when to use* keys (Peterka, [0116]-[0121]) are *complementary* to Peterka's pull model for acquiring keys (Peterka, [0105]-[0106]) and do not require use of Peterka's push model. As discussed previously in paragraph 49, the concepts of when to use keys and when to acquire keys are separate and orthogonal. Peterka's teachings regarding signaling when to use new keys *do not* require Peterka's server to actively keep track of which clients have which keys or *push* updated keys to clients without first receiving a client request.⁶ The fact that a pull model (in which clients monitor key expirations

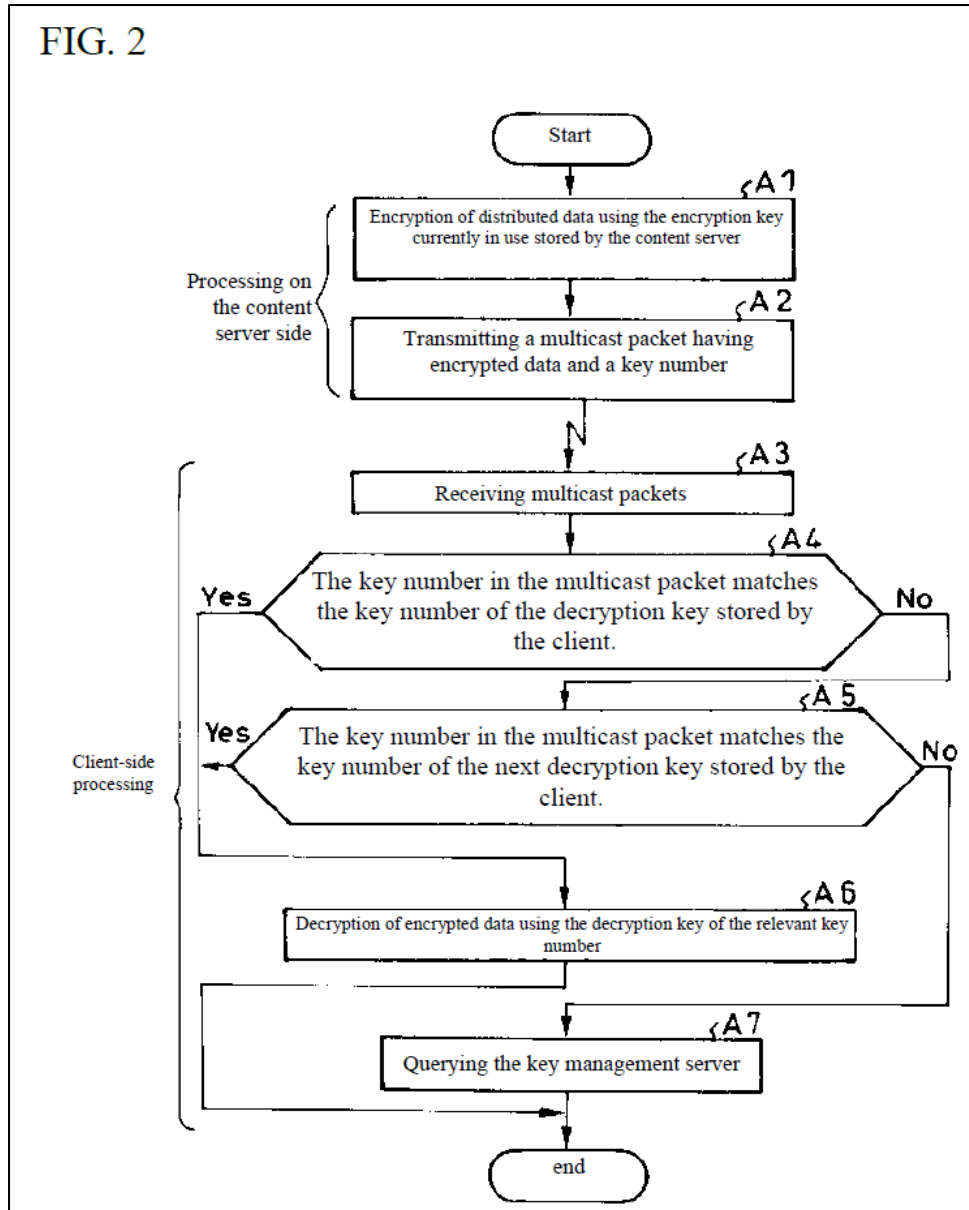
⁶ In the POPR of the IPR2022-00618 proceeding, in discussing Peterka, Patent Owner asserted "[i]n the embodiments of paragraphs 116-121, the client thus *already has* the new key, and does not issue any request for a new key, as limitation 1.5 of the '178 Patent requires." EX-1027, at page 19 (emphasis in original). In my opinion this assertion is either a disingenuous representation of Peterka's disclosures, or it demonstrates a tortured misunderstanding of same. As I've discussed, the POSITA would have well understood that Peterka's key distribution teachings under "a 'Pull' model, a 'Push' model, and a combination of 'Push-Pull' model" in paragraph [0105] are separate from Peterka's teachings regarding how clients know when to apply specific keys, irrespective of whether keys were pushed or pulled.

and request keys as needed) is compatible with key change signaling in the content stream is *confirmed* by Peterka's express teaching to provide *both* "the time remaining in the lifetime of the key" and the "key parity bit" in the ECM (or EMM) that provides the content key: "Another field can be provided for the time remaining in the lifetime of the key. *Furthermore*, a key parity bit matching the parity bit in the RTP packet can be provided." Peterka, [0121]. This disclosure in Peterka confirms to the POSITA that the two concepts are *not* mutually exclusive, because the POSITA understood (1) that the "time remaining in the lifetime of the key" was used to monitor when keys would expire and allowed the client to request new keys before the next key rotation boundary, and (2) that the "key parity bit" allowed the client to determine which key to apply for decryption (and when).

116. For additional corroborating documentary evidence that a pull model for key distribution was compatible with content stream signaling, see the Suzuki reference, which describes a client that uses a pull system for key distribution *and also* detects signals for key changes in packets of the content stream. *See, e.g.,* Suzuki, [0016] (client both "transmits a key information request to the key management server when the remaining validity time of the key in use" approaches 0, *and also* "switches the key that was previously stored in the local table as the next key to be used to the new key in use *based on the recognition of changes in key numbers in multicast packets sent from content servers*"), [0013] (similar), [0059]-

[0060] (describing client sending key information requests to obtain keys), [0076]-[0077] (same), [0043]-[0047] (describing client comparing key number in content stream with key numbers of keys stored at the client to determine whether (and when) to switch keys for decryption), FIG. 2 (reproduced below, showing that the client uses signals in the content stream—*i.e.*, a key number in the same multicast packet that also includes Suzuki's encrypted content—to determine which key to apply for decryption).

FIG. 2



See also Suzuki, FIG. 5 (reproduced below, showing that Suzuki's system provides keys in response to "key information request[s]" (81) from a client and provides "key number" in "multicast packet[s]" from content server (21), to signal key changes to clients).

Figure 5

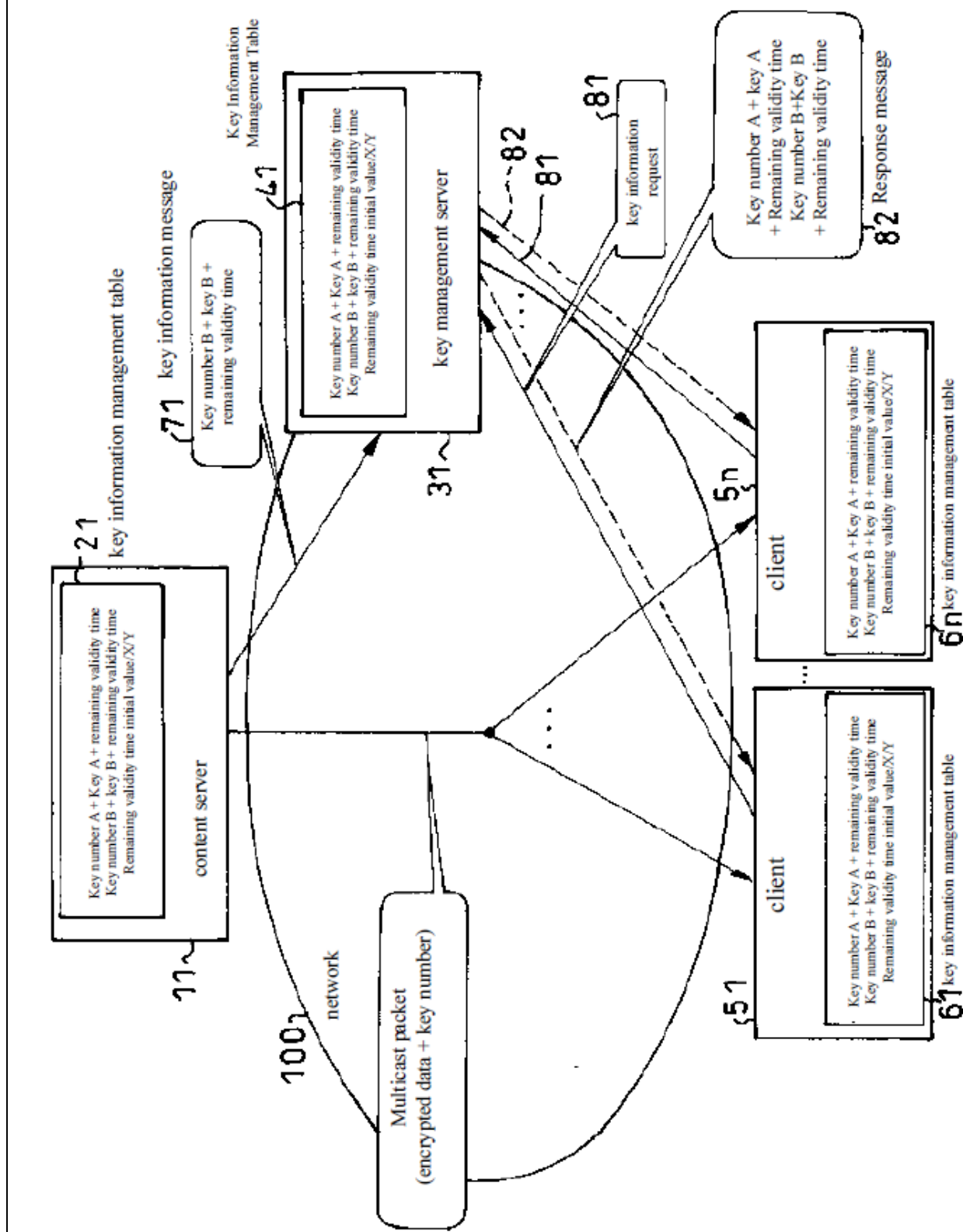


Figure 5 of Suzuki above illustrates a system that uses the pull model for key distribution as that phrase is used in Peterka (*e.g.*, Peterka, [0105]). Specifically, Suzuki shows a client (51) that keeps track of its keys: it monitors when its keys expire (using “remaining validity time” for each key) and submits a key information request (81) to a key management server (31) before its current key expires. *E.g.*, Suzuki, [0013], [0016], [0059]-[0060], [0076]-[0077]. Suzuki’s servers ***do not keep track*** of which client has which keys, and keys are transmitted by a server ***only*** after a client requests them. *Id.* ***The same system*** in Suzuki (see Figure 5) ***also uses*** signals directly in the content stream. This is shown in Suzuki’s multicast packets from the content server (11), which include both encrypted content ***and a key number***, which the client uses to determine which of the keys in its possession must be applied to decrypt the encrypted content. *E.g.*, Suzuki, [0013], [0016], [0043]-[0048]. Suzuki’s system confirms that a pull model for key distribution is fully compatible with using key change signals in the content stream, and that such systems were well-known and well-documented at the time of the alleged invention.

117. At minimum, using Peterka’s key change signaling (Peterka, [0116]-[0120]) with Peterka’s pull model (Peterka, [0105]-[0106], [0110]) would have been obvious in view of (1) Peterka’s express teachings (*e.g.*, Peterka, [0121], as described above in paragraph 115) and (1) a POSITA’s background knowledge (which is corroborated by the Suzuki reference, as described above in paragraph

116). Such an implementation of Peterka would have been desirable because clients would know precisely when to start applying new keys without burdening Peterka's servers with the responsibility of keeping track of each individual client's keys (and their expirations). Additionally, the POSITA would have reasonably expected success with such an implementation given that systems using both a pull method for key distribution and key change signaling in the content stream were well-known and documented, as evidenced by Suzuki.

- vi. **[1E] “issuing requests to a license server ahead of a key rotation boundary to retrieve a second content encryption key to be used after a content encryption key rotation boundary is reached; and”**

118. Peterka discloses a client device issuing at least one request ahead of a key expiration time (*ahead of a key rotation boundary*) to retrieve a new key (*retrieve a second content encryption key*)⁷ that is, as discussed below in

⁷ I understand that in the 1376 Investigation, the ITC agreed with Patent Owner that Elements [1D] and [1E] (and Elements [16E] and [16F]) “do not need to be performed in the recited order” that they appear in the claims. EX-1032, 21-25. My conclusion that Peterka invalidates the claims would not change even if the claims were construed to require Elements [1D] and [1E] (and Elements [16E] and [16F]) to occur in the order that the respective elements are recited in the claims. I note

§ VIII.A.1.a.vii, *to be used after a content encryption key rotation boundary is reached*. Peterka, [0105]-[0106] (“Under the pull model, each client keeps track of the keys and their expiration times and actively *requests* new keys before the current keys expire so as to avoid service interruptions.”). For additional context, see my discussion above in §§ VIII.A.1.a.ii-VIII.A.1.a.iii.

119. Peterka discloses issuing such requests to a “key distribution server,” which a POSITA understood to be a *license server*. Peterka, [0110] (“Under a pull key distribution model, each client requests a new key and is individually responded to by *the key distribution server*.”); ’178 Patent, 3:44-45 (explaining that a “license server is responsible for *distributing* content encryption *keys to clients*”).

120. The claim appears to recite *multiple* “requests to a license server... to retrieve a second content encryption key,” though the ’178 specification only appears to disclose a client issuing a single request to a license server for each new key that it needs. ’178 Patent, 11:21-25 (“client 110 must issue a new content

that, at the very least, Peterka’s timer-based detection (which I described above in VIII.A.1.a.v) indisputably occurs before and indeed facilitates Peterka’s requests “to retrieve a second content encryption key.” For at least that reason, Peterka would meet the claims even if Elements [1D] and [16E] needed to respectively occur before Elements [1E] and [16F].

encryption key request to the license server 106 to retrieve the new content encryption key”), 11:3-62 (“If the new content encryption key has not been retrieved yet (e.g., due to early expiration of the current content encryption key for security reasons, or content encryption key retrieval failures in step 318), the client 110 must issue a new content encryption key request to the license server 106.”), FIG. 3; *see also* ’178 Patent, 1:45-50 (“[E]ach content encryption key is provided upon expiration of a period of use of a serially preceding content encryption key.”), 9:63-65 (describing two requests to obtain a new key, but only one to a license server). Notably, I am informed that in the 1376 ITC Investigation, Patent Owner interprets Element [1E] to encompass issuing one license request for each subsequent key that is needed, consistent with the specification. *See* EX-1033, pages 14-15.

121. A POSITA would have understood Peterka—in which a client issues a series of (multiple) requests for keys to decrypt upcoming media segments during a stream—meets this limitation at least to the same extent it is described in the ’178 Patent. Peterka, [0080]-[0082], [0105]-[0106].

122. Additionally, clients participating in media distribution systems commonly re-issued requests to a server if a first request failed due to a timeout, a server error, or transmission failure. *See, e.g.* EX-1019 (RFC793: Transmission Control Protocol), at pages 4 (“If the [positive acknowledgment] is not received within a timeout interval, the data is retransmitted.”), 40 (“Because segments may

be lost due to errors (checksum test failure), or network congestion, TCP uses retransmission (after a timeout) to ensure delivery of every segment.”), 77 (“For any state if the retransmission timeout expires on a segment in the retransmission queue, send the segment at the front of the retransmission queue again, reinitialize the retransmission timer, and return.”); EX-1020 (RFC2616: Hypertext Transfer Protocol – HTTP/1.1), at page 43 (“The client did not produce a request within the time that the server was prepared to wait. The client MAY repeat the request without modifications at any later time.”). To ensure that a client would have a next key available in the event of a failed request, it would have been obvious to implement Peterka to re-issue a request for a second content encryption key if a first request failed for any reason, and a POSITA would have reasonably expected success with such an implementation. *See, e.g.*, EX-1019, at pages 4, 40, 77; EX-1020, at page 43. Peterka thus discloses (or renders obvious) Element [1E].

vii. [1F] “applying the second key for content decryption after the key rotation boundary is reached.”

123. As discussed above in § VIII.A.1.a.vi, Peterka discloses its client actively requesting a new key (*the second content encryption key* in [1E]) from a key distribution server before the current key expires (*the key rotation boundary*), to avoid service interruptions. Peterka, [0101], [0105]-[0106], [0110]. Based on my review of the claim language and specification, I understand (and a POSITA would

have understood) that the claim encompasses an interpretation where the *second key for content decryption* recited in [1F] is the same *second content encryption key* recited in [1E]. I have applied that interpretation below.

124. Peterka discloses *applying* the new key it requested (in other words, *the second key for content decryption*) *after the key rotation boundary is reached*—for example, after the previous key expires. Peterka, [0106] (disclosing that “decryption by the client” occurs “by utilizing the cryptographic key that was requested” prior to the previous key’s expiration), [0116] (“[T]he content key is utilized for decrypting the program content throughout the course of the distribution of the program content.... [W]hen a new content key is implemented, the implementation will be signaled to the clients so that the clients can begin utilizing the new content key with which they have been provided”), [0090] (disclosing a client that “receive[s] keys for decrypting the encrypted portion of the program”), [0092] (explaining that “keys are required to decrypt the remainder of the program content”), [0108] (disclosing that “the program content is distributed for decryption by the client utilizing the transmitted key”). For additional context, see my discussion above in § VII.A.

- b. **Claim 2:** “The method of claim 1, wherein the content encryption keys and content encryption key identifiers returned in the playback request response include the content encryption key and associated identifier currently being applied as well as a future content encryption key and associated identifier yet to be applied.”

125. As described above in §§ VIII.A.1.a.ii and VIII.A.1.a.iii(2), Peterka discloses a client receiving multiple *content encryption keys*—for a “*current*” and “next” (*future*) segment, each in one ECM or EMM—in the *playback request response* described above in § VIII.A.1.a.ii. Peterka, [0080] (“keys to the current as well as the next segment”), [0130] (describing FIG. 7B, which illustrates content keys for current and future segments sent when accessing a stream during a current segment), FIG. 7B. For the reasons described above in § VIII.A.1.a.iii (e.g., in paragraphs 90-96), a POSITA would have understood Peterka to include (or would have found obvious to implement Peterka to include) identifiers corresponding to those keys (*associated identifiers*). Thus, Peterka meets the requirement that “*the content encryption keys and content encryption key identifiers returned in the playback request response include the content encryption key and associated identifier currently being applied as well as a future content encryption key and associated identifier yet to be applied,*” as claimed.

- c. **Claim 3: “The method of claim 1, wherein the content encryption key expiration returned in the playback request response is expressed as an expected minimum interval between periodic key rotation.”**

126. Peterka discloses media streams that are divided into “purchasable segments” to allow users to pay according to the amount of time—*e.g.*, number of segments—that the user consumes. Peterka, [0078]-[0082] (discussing Peterka’s “Pay-by-time (PBT)” mechanism), [0099]-[0100] (describing one way to divide a program into purchasable segments), [0101] (describing Peterka’s content keys). As discussed above in §§ VII.A and VIII.A.1.a.iii, each segment is encrypted by a different content key.

127. The POSITA would have understood Peterka to also indicate that segments have a defined *period* that a POSITA would have understood corresponds to *periodic key rotation*. See Peterka, [0081] (“viewer will be told what the pay *periods* are”), [0082] (“client monitor[s] the expiration time for each pay *period*” and correspondingly “request[s] a new set of keys from the” server); *see also* Peterka, [0110] (explaining that “[u]nder a pull key distribution model,” client is “*periodically* given new keys” and “server... issue[s] new keys at *regular* intervals”).

128. The ’178 specification leaves the implementation details of how to “*express[]*” an “expiration” “as an *expected minimum interval* between periodic key

rotation” to the POSITA. Indeed, the specification does not appear to use the terms “expected,” “minimum,” or “interval” at all—alone or together.

129. As I discussed above in § VIII.A.1.a.iii [subsection c)], Peterka discloses that each key distribution message (*e.g.*, an ECM or EMM) includes a field for the “***time remaining*** in the lifetime of the key” provided in the message. Peterka, [0082], [0105]-[0106], [0121], FIG. 8. For additional context, see my discussion above in § VIII.A.1.a.iii. Peterka thus teaches key expiration times expressed as a remaining amount of time in the key’s lifetime—*e.g.*, number of minutes/seconds (an interval)—instead of a specific time-of-day at which the key expires (*e.g.*, 12:00am). Peterka, [0121].

130. Given Peterka’s disclosure of periodically-distributed keys that are issued “at regular intervals” (Peterka, [0082], [0110]), a POSITA would have understood that the disclosed “expiration time” included in each ECM or EMM (Peterka, [0121]) constitutes a “***minimum*** interval” that the corresponding key (in the ECM or EMM) is “***expected***” to be used “***between***” periodic key changes. A POSITA would have understood that Peterka’s key would not have been “expected” to be used for ***less*** than the time remaining in the key’s lifetime before the next key change, because using a key for ***less*** than the time remaining in the key’s lifetime—such that a client stopped using a key before it expired and rotated the key before a next key was used to encrypt media content—would have resulted in service

interruptions caused by the client (on its own) trying to apply a next key to decrypt content that had been encrypted using the current key. A POSITA would have understood that the expiration time of each key in Peterka is thus an expression of the ***minimum interval*** of time that the key is ***expected*** to be used, and a POSITA would have understood Peterka’s expiration times are ***expressed as an expected minimum interval between periodic key rotation***, as claimed.

131. Alternatively, this would have been obvious. The question of exactly how to “express[]” key “expiration”—e.g., as an expected ***minimum*** interval (between periodic key rotation), an expected ***maximum*** interval, or both—is a design choice that was well within the skill of a POSITA to implement. Moreover, this choice is not described at all in the specification or assigned any importance. The specification notably does not even use the terms “expected,” “minimum,” or “interval” at all.

132. For example, a POSITA would have had reasons to use (***express***) Peterka’s “expiration time[]”/“time remaining in the lifetime of the key” (Peterka, [0082], [0105]-[0106], [0121]) as both an ***expected minimum interval between periodic key rotation*** and an expected maximum, such that keys were expected to expire and rotate on a fixed, periodic basis that defines a minimum (and maximum) interval between periodic key rotation—and would have reasonably expected success doing so. This would have predictably helped ensure that keys were used

by the client in sync with the key rotation occurring at the encrypting server, and helped increase operational robustness and security by ensuring clients could not use keys after their expected expirations.

133. Moreover, a POSITA would have had reasons to use (*express*) Peterka’s disclosed “expiration time[]”/“time remaining in the lifetime of the key” (Peterka, [0082], [0105]-[0106], [0121]) as only an *expected minimum interval between periodic key rotation*—as opposed to an expected *maximum* interval, where a client would be prohibited from applying a key in its possession after the time remaining in the key’s lifetime goes to zero—and would have likewise reasonably expected success doing so. This would have, for example, ensured that keys were used by the client in sync with the key rotation occurring at the encrypting server, and also would have provided flexibility, allowing Peterka’s clients to continue using an already-acquired key in scenarios where it may be deemed unnecessary to rotate the key (*e.g.*, when a final key rotation boundary occurs near the end of a program). While this may have allowed clients to view a limited amount of content (*e.g.*, a program’s end credits) using an expired key, a POSITA would have reasonably expected success as it still would have provided the security associated with key rotation but also helped provide uninterrupted service (operational robustness) to authorized clients.

134. At the very least, either option—two of three finite options (expirations expressed as an expected minimum interval, an expected maximum interval, or as both an expected minimum interval and an expected maximum interval)—would have been obvious to try.

d. Claim 4: “The method of claim 3, further comprising: timing the prefetching of a next un-retrieved content encryption key based on an expected expiration of the content encryption key currently being used.”

135. The ’178 specification uses “prefetching” to encompass acquiring a key before it is needed. ’178 Patent, 11:12-18 (“The client 110 first checks whether the content encryption key corresponding to a key change notification in step 308 has already been retrieved by prefetching as described below with reference to step 318. If the new content encryption key has already been retrieved, the current content encryption key is replaced with that pending content encryption key.”), 11:48-53 (“If the current content encryption key is within a fixed threshold of expiring or a content encryption key change request was detected in step 308, processing proceeds to step 318 where the client 110 requests a new content encryption key from the license server 106.”).

136. As discussed above, Peterka discloses its client “keep[ing] track of”/“monitor[ing]” key expiration times and requesting a new key (*a next un-retrieved content encryption key*) before the new key is needed (*prefetching*), based on when (and before) “current keys expire” (*based on an expected expiration of the*

content encryption key currently being used). Peterka, [0105]-[0106] (“Under the pull model, each client keeps track of the keys and their expiration times and actively requests new keys before the current keys expire so as to avoid service interruptions.”), [0082] (“The client will monitor the expiration time for each pay period and request a new set of keys from the caching server.”). For additional context, see my discussion above in §§ VIII.A.1.a.v-VIII.A.1.a.vi and VIII.A.1.c.

- e. **Claim 5: “The method of claim 4, further comprising: prefetching a next un-retrieved key a fixed duration before the expected expiration of the content encryption key currently being used.”**

137. Claim 5 depends from claim 4, and requires that a “prefetching” of “a next un-retrieved key” occurs at “a fixed duration before the expected expiration” of the currently-used key.

138. A POSITA understood that there were a limited number of ways to implement precisely *when* Peterka’s client acquires new keys. One conventional, straightforward way was to implement Peterka’s client—which, as discussed above, already (1) receives expiration times expressed as the “time remaining” in a key’s lifetime (Peterka, [0121]), (2) “keeps track of”/“monitor[s]” key “expiration times” (Peterka, [0105]), and (3) “actively requests new keys before the current keys expire so as to avoid service interruptions” (Peterka, [0105])—was to have the client request new keys a specific amount of time (a *fixed duration*) before the time

remaining in the key's lifetime reached zero ("before the current keys expire" (Peterka, [0105])).

139. For corroboration of a POSITA's background knowledge that new keys in a system with rotating keys could be acquired a specific amount of time before key expirations, see, for example, the Suzuki reference. *E.g.*, Suzuki, [0036] ("FIG. 4 is a flow chart showing processing relating to key management in the key exchange method in the multicast distribution system according to this embodiment (processing at the point in time when the remaining validity time of the key in use reaches the second setting value)."), [0059] ("When the remaining validity time of the key in use in the key information management table 6i of the client 5i reaches the second setting value, the key management means 51 in the client 5i confirms to that effect (that the remaining validity time value has decreased and reached the second setting value) (step C1), and to obtain key information relating to the next key (decryption key) to be used, a key information request 81 is sent to the key management server 31 (step C2)."), [0076]-[0077] ("When the remaining validity time of key A reaches Y (second setting value), the client 51 transmits a key information request 81 to the key management server 31, requesting key information on the next key to be used."), FIGs. 4-6.

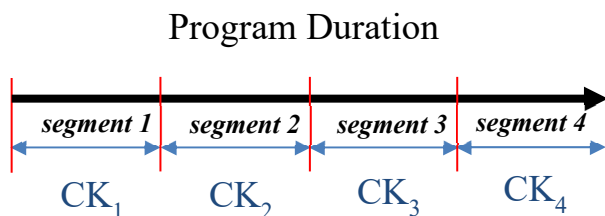
140. Moreover, a POSITA would have appreciated that consistently acquiring new keys at a specific time (a *fixed duration*) before the current keys

expire was a way “to avoid service interruptions” (Peterka, [0105]) because this would have helped ensure new keys were requested with sufficient time for servers to respond to large numbers of client key requests prior to key rotation boundaries. Additionally, a POSITA would have understood that by giving clients different fixed durations, prefetching could be distributed in time, balancing the server load. Implementing clients to consistently acquire new keys a (customized) fixed duration before key expirations would have, for example, helped predictably, and with a reasonable expectation of success, account for server delay—that is, delays that occur while processing client requests. Such a strategy would also have, for example, helped predictably, and with a reasonable expectation of success, account for network latency—that is, delays that occur when data travels across a network.

- f. **Claim 6: “The method of claim 4, further comprising: prefetching a next un-retrieved key within a period of time after the expected expiration of the content encryption key currently being applied, the period of time beginning at the expected expiration of the content encryption key currently being used, and ending at a duration calculated as a fixed percentage of a fixed periodic content encryption key expiration interval.”**

141. Claim 6 depends from claim 4, and requires that a prefetching of “a next un-retrieved key” occurs *after* the current key expires (“within a period of time after the expected expiration of the content encryption key currently being applied”).

142. Peterka’s system, which meets claim 5 as discussed above in § VIII.A.1.e, also meets claim 6. The diagram below—depicting a content stream divided into segments of equal duration and encrypted by a series of content keys (CK₁, CK₂, etc.)—helps illustrate why.



143. In Peterka, a client acquires CK₂ during segment 1, *before* CK₁ expires. Peterka, [0105]-[0106] (a client “actively requests new keys before the current keys expire”). A client later acquires CK₃ during segment 2, *after* CK₁ but *before* CK₂ expires. *Id.* A POSITA understood that both CK₂ and CK₃ are *prefetched* because they are acquired before they are needed. ’178 Patent, 11:12-18 (“The client 110 first checks whether the content encryption key corresponding to a key change notification in step 308 has already been retrieved by prefetching as described below with reference to step 318.”), 11:48-53 (“If the current content encryption key is within a fixed threshold of expiring or a content encryption key change request was detected in step 308, processing proceeds to step 318 where the client 110 requests a new content encryption key from the license server.”).

144. I note that Claims 4-6 each recite “a next un-retrieved key.” However, claims 5-6 *do not* recite the *same* “next un-retrieved key” or “the next un-retrieved

key” that was recited in claim 4. Thus, a POSITA would have understood that claim 5 encompasses a client acquiring (“prefetching”) CK₂ (“a next un-retrieved key”) during segment 1 (*before* the current key, CK₁, expires), and that claim 6 encompasses acquiring (“prefetching”) CK₃ (a different “next un-retrieved key”) during segment 2 (*after* the current key, CK₁, expires).

145. As discussed for claim 5 (above in § VIII.A.1.e), there were a limited number of ways to implement when Peterka’s client acquires new keys, of which one was a “fixed duration” before the current key’s expiration, as is confirmed by the Suzuki reference, which I cited to corroborate a POSITA’s background knowledge. In such an implementation (with reference to the diagram and example described above), that would have entailed acquiring CK₂ a *fixed duration* before the current key CK₁’s expiration. This would have satisfied claim 5.

146. Additionally, in such an implementation, a POSITA would *also* have understood that CK₃ was acquired “*within*” “*the period of time beginning at the expected expiration of the content encryption key currently being used,*” (*after* CK₁’s expiration) “*and ending at*” a fixed duration *before* the next key rotation boundary (*before* CK₂’s expiration). Acquiring CK₃ at a fixed duration *before* CK₂’s expiration—which, for Peterka’s key rotation period (Peterka, [0082], [0110]), would be a fixed percentage of the key rotation period—constituted *a duration*

calculated as a fixed percentage of a fixed periodic content encryption key expiration interval, as claimed in claim 6.

147. Alternatively, a POSITA would have had reason to implement Peterka to time the “prefetching” of a new key (discussed above in § VIII.A.1.d) based on a *fixed percentage of a fixed periodic key expiration interval*, which would have met claim 6. Such an implementation would have enabled Peterka’s clients to seek new keys for every key rotation period, no matter how short the rotation period was, and would have ensured that clients could prefetch keys even for high-security media streams in which keys were rotated very quickly. This is because, by basing a client’s prefetch time on a *fixed percentage* of the key rotation period (a *fixed periodic key expiration interval*), the key rotation period could be reduced (or indeed raised) without requiring any additional modifications to ensure that requests were being made prior to the key expiration. A POSITA would have reasonably expected success with such an implementation.

- g. Claim 10: “The method of claim 1, wherein license server communications are secured and authenticated using a selected one of Secure Sockets Layer and a token-based technique using a token encrypted with a symmetric key.”**

148. Claim 10 requires “license server communications” to be “secured and authenticated using a *selected one*” of two methods for securing communications with a license server: (1) “Secure Sockets Layer” and (2) “a token-based technique

using a token encrypted with a symmetric key.” The ’178 Patent specification does not describe any embodiment that uses both methods, and the claim lists no other alternatives. Thus, I understand that only one listed alternative must be satisfied to meet the claim. As discussed below, both alternatives were obvious.

149. Secure Sockets Layer (“SSL”) was a well-known standard for securing communications channels using encryption. A copy of the standard is included in this proceeding as EX-1021. SSL was used by numerous websites in (and before) 2011.

150. Peterka discloses its client-server communications occurring over the Internet. Peterka, [0004] (“[A] server can multicast a movie across the internet to client computers.”), [0015] (“FIG. 1 is a block diagram of an embodiment of a content distribution system such as those used for multicasting program content over the internet.”), [0038] (describing an embodiment of the content distribution system that includes “a network such as the internet”), [0135] (“System 1900 shows a client server network comprised of at least one client 1908 coupled to a server such as origin content processor 1904 via a network 1916 such as the internet.”). A POSITA would have had reason to implement Peterka’s client-server communications over the Internet using SSL, as this was a prevailing standard for Internet-based communications, and a conventional, established, predictable option for preventing

eavesdropping, tampering, or message forgery. Given the standard's ubiquity, a POSITA would have reasonably expected success with such an implementation.

151. Peterka also discloses the claimed “token-based technique.” A POSITA understood the claimed *token-based technique* to encompass exchanging encryption key information in an encrypted form (*token*). See, e.g. '178 Patent, 6:57-58 (“In one embodiment, the information is stored as an encrypted token in a database.”). For documentary evidence corroborating a POSITA's knowledge regarding the well-known concept of using tokens in conjunction with encryption, see, for example, the Barrus (EX-1022) reference. E.g., Barrus, 1:26-39 (“A token comprises information encoding an identifier and an encryption key.”), 5:18-25 (“A unique encryption key is then generated/identified for the token to be generated (step 210).”).

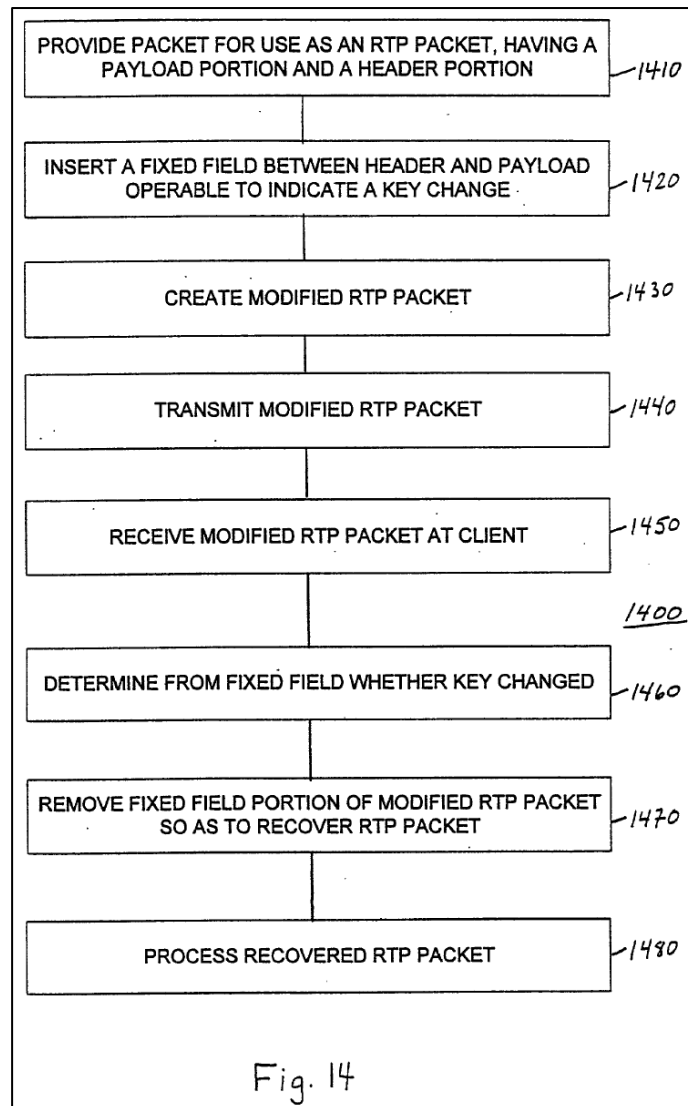
152. Peterka discloses a framework for securing communications between clients and license servers by requiring clients to present a security “ticket” to the server when requesting access to content or fetching keys from the server. Peterka, [0066] (“The client delivers the secure object and a ticket with its own entitlement data... to the caching server.... If all [content access] rules are satisfied, the caching server will grant access to the requested content.”), [0075] (“The caching server will compare the information from the secure object and the ticket. If the information matches, the client will be granted access to the content (it will be given the content

encryption key—delivered directly or indirectly utilizing a Service Key in this case).”), [0136] (“In one embodiment of the invention, a client registers with the authorization center 1920 to obtain a ticket which defines what type of content the client is entitled to obtain.”), [0142]-[0144] (“This ticket is comprised of entitlement information that can be use[d] to determine whether the client is entitled to receive the program content.”), FIGs. 19-21. Peterka explains that the client ticket or similar data is “authenticated and encrypted so that the client cannot revise his own entitlements.” Peterka, [0146], FIG. 23. The client ticket thus provides an encrypted security “token” allowing the server to authenticate a client and determine whether to grant a request for keys to access streaming media content.

153. Peterka does not expressly detail whether its client ticket (or other token) is encrypted with a *symmetric key*, though this would have been an obvious option because “symmetric” encryption was one of two common encryption options at the time of the alleged invention, and one that did not require the creation of public-private key pairs or the maintenance of more than one key. For documentary evidence corroborating this background knowledge of a POSITA, see the Kelly (EX-1010) reference. *See* Kelly, [0004] (“Examples of cryptographic techniques include *symmetric encryption*, *asymmetric encryption* and hashing.”). A POSITA would have reasonably expected success implementing Peterka to use symmetric keys to encrypt its client ticket (or other token).

h. Claim 12: “The method of claim 1, wherein actual content encryption key rotation boundaries are detected based on real-time in-band notifications.”

154. As I discussed above in § VIII.A.1.a.v (*e.g.*, in paragraphs 114-115), Peterka discloses its client checking content packets for a signal to start applying a new key, which meets Patent Owner’s narrow interpretation of the “detect” limitation. Peterka, [0116]-[0121] (describing Peterka’s “signaling methods”); *see also* Peterka, FIG. 14 (showing one way Peterka teaches including a signal in the content stream).



155. The POSITA understood that Peterka’s disclosed signal-based detection was “based on real-time in-band notifications,” because the described signals are embedded in Peterka’s content packets (*in-band*) and occur when the client needs to switch keys for decryption (in *real-time*). See ’178 Patent, 4:13-15 (“The packager is responsible for providing in-band notification to the client for the key change.”). The POSITA likewise understood that this constitutes detection of

“*actual* content encryption key rotation boundaries,” as claimed, because a different key is *actually* needed for decryption before and after the client receives the signal.

i. Claim 16

i. [16PRE] “A computerized device operable as a client for handling secure distribution of content, comprising:”

156. Peterka discloses a client device (*computerized device*) operable as a client for handling secure distribution of content. Peterka, [0038]-[0042], FIG. 1 (“client computer 112”), FIG. 2 (describing “individual system elements” of devices including client computer 112). For additional context, see my discussion above in § VIII.A.1.a.i.

ii. [16A] “memory operative to store computer program instructions; one or more processors; input/output interface circuitry; and interconnect circuitry coupling the memory, processors and input/output interface circuitry together,”

157. Peterka discloses how “individual system elements from FIG. 1,” which includes “client computer 112,” can be implemented with system components like those depicted in FIG. 2. Peterka, [0041], FIG. 2; *see also* Peterka, [0038]-[0043], FIGs. 1-2. Peterka expressly discloses that such a system (*e.g.*, client computer 112) can include memory operative to store computer program instructions (*e.g.*, working memories 209/291/292/293), one or more processors (*e.g.*, CPU(s) 201, processing acceleration 207), input/output interface circuitry (*e.g.*, I/O devices

202/203), and interconnect circuitry coupling the memory, processors, and input/output interface circuitry together (*e.g.*, bus 208). *Id.*

iii. [16B]-[16G]

158. Peterka further provides that the processors are operative to execute the computer program instructions from the memory to cause the client device to perform the operations described in Elements [16B]-[16G], for the same reasons described above in the context of Elements [1A]-[1F], as summarized in the table below.

Claim 16 Element	Corresponding Claim 1 Element	Section
[16B]	[1A]	§ VIII.A.1.a.ii
[16C]	[1B]	§ VIII.A.1.a.iii
[16D]	[1C]	§ VIII.A.1.a.iv
[16E]	[1D]	§ VIII.A.1.a.v
[16F]	[1E]	§ VIII.A.1.a.vi
[16G]	[1F]	§ VIII.A.1.a.vii

j. Claims 17-18, and 20

159. Peterka's computerized device further satisfies the requirements recited in Claims 17-18 and 20 for the same reasons described above in the context of Claims 2-4 and 12, as summarized in the table below.

System Claim	Corresponding Method Claim	Section
17	2	§ VIII.A.1.b
18	4	§ VIII.A.1.d
20	12	§ VIII.A.1.h

B. Ground 2: Peterka in View of Suzuki Renders Claims 1-6, 10, 12, 16-18, and 20 Obvious

160. To the extent the encryption key “identifiers” recited in Element [1B] or the timing requirements for the “prefetching” recited in claims 4-6 are not considered disclosed or rendered obvious by Peterka alone, Suzuki expressly teaches those limitations. As discussed below, modifying Peterka in view of Suzuki separately renders obvious claims 1-6, 10, 12, 16-18, and 20.

1. Peterka-Suzuki

161. As I discussed above, for example, in §§ VII.A and VIII.A.1.a.v, Peterka describes clients acquiring “keys” for multiple media segments, “keep[ing] track of”/“monitor[ing]” keys “and their expiration times,” and “determining when a new key is required.” Peterka, [0080]-[0082] (“The client will monitor the expiration time for each pay period and request a new set of keys from the caching server.”), [0105]-[0106] (“Under the pull model, each client keeps track of the keys and their expiration times and actively requests new keys before the current keys expire so as to avoid service interruptions.... Thus the server need not monitor the status of the client’s keys, rather, the responsibility of determining when a new key is required can be passed to the client.”), [0114] (“[The client] can monitor the expiration time of the program segment key and send an individual key update request before the key expires.”). Peterka leaves to the POSITA how to keep track of keys and their expiration times, and how to determine *when* a new key is required.

Suzuki teaches such details, describing an implementation of Peterka's disclosed functionality using "key identifiers" for keeping track of keys, and using a client's clock signal to request a new key a fixed amount of time before the current key expires. Suzuki, [0071]-[0084].

162. Similarly to Peterka, Suzuki teaches providing a client with "key information" regarding a "key currently in use (key A) and the next key to be used (key B)." Suzuki, [0072]-[0075] ("FIG. 5 is a diagram specifically showing the information in each key information management table and the contents of each packet/request/message. In FIG. 5, the key currently in use is indicated as key A, the next key to be used is indicated as key B.... FIG. 6 is a diagram showing an operation sequence over time when the information shown in FIG. 5 is handled."), FIGs. 5-6. Suzuki teaches clients acquiring this information by submitting a "key information request" to a key management server, which responds with a "key response message" that includes (1) the current and next keys ("key A," "key B"), (2) each key's "key number" ("key number A," "key number B"), and (3) each key's "remaining validity time." Suzuki, [0076]-[0077] ("When the remaining validity time of key A reaches Y (second setting value), the client 51 transmits a key information request 81 to the key management server 31, requesting key information on the next key to be used. In response to this key information request 81, the key management server 31 transmits a response message 82 having key information

relating to the key currently in use (key A) and the next key to be used (key B). As a result, the client 51 has key information relating to the keys A and B, as shown in FIG. 5.”), FIGs. 5-6. As discussed further below, Suzuki includes teachings regarding how to use its “remaining validity time” and “key number.”

163. “Remaining validity time” is used to determine when to request new keys. Suzuki, [0076]-[0077]. To “eliminat[e]... confirmation delay” during key changes—*e.g.*, due to client-server “time lag” (Suzuki, [0008])—Suzuki teaches having the client submit a “request” for a new key when “the remaining validity time” of the current key “reaches” a specific, greater-than-zero “setting value.” Suzuki, [0076]. The client calculates this “setting value” as a “certain percentage of the time from the initial value” of the “remaining validity time of the key.” Suzuki, [0070]. The client knows when this “setting value” is reached by “updat[ing] the remaining validity time” for each key in its possession using the client’s “clock signal[]” and performing “processing” that includes “recogni[zin]g” that “the value of the remaining validity time has decreased and reached a [particular] setting value.” Suzuki, [0033] (“The content server 11, the key management server 31, and each client 5i update the remaining validity time in the key information in the key information management table 21, the key information management table 41, and each key information management table 6i that they manage every time the time passes (performing updates based on their own clock signals, and the like).”), [0036]

(describing FIG. 4, which shows processing relating to the key exchange), [0059] (also describing FIG. 4: “When the remaining validity time of the key in use in the key information management table 6i of the client 5i reaches the second setting value, the key management means 51 in the client 5i confirms to that effect (that the remaining validity time value has decreased and reached the second setting value) (step C1), and to obtain key information relating to the next key (decryption key) to be used, a key information request 81 is sent to the key management server 31 (step C2). In addition, the address of the key management server 31 to which the key information request 81 is to be sent is preset in the client 5i.”), FIG. 4. Thereby, a client can request and “obtain key information regarding the next key to be used *before the key is changed.*” Suzuki, [0080]-[0081].

164. “Key number” is used to “identify[] the key,” allowing a client to associate keys with their respective “remaining validity time[s]” (e.g., in a table) and to help the client keep track of keys, for example when determining which key to apply when decrypting a segment of content. Suzuki, [0030] (“The client 5i has a key information management table 6i that stores a pair (key information) of a key (decryption key) used to decrypt data distributed by multicast packets, a key number for identifying the key, and the remaining validity for the key.”), [0043]-[0048] (“If it is determined in step A4 that ‘the two key numbers match,’ the encrypted data in the multicast packet is decrypted using the decryption key of that key number (step

A6).”), [0080] (“[W]hen the key management server 31 receives key information on the key C, which is the key to be used next after the key B, in a response message 82 from the key management server 31, the key management server 31 stores the key information in its own key information management table 6i.”), FIG. 2. A POSITA would have recognized the importance of such identifiers to differentiate amongst keys, and that providing unique values (such as key numbers) to keys would ensure that the system does not confuse keys.

165. Peterka and Suzuki are in the same field of art. Like Peterka (as discussed above at § VIII.A.1.c), Suzuki discloses “periodically” changing keys that protect distributed content. Suzuki, [0003] (“At this time, by periodically changing the key, it is possible to manage clients and further improve the confidentiality of the encryption.”), [0012] (“[T]he feature of the present invention is that when encryption of distribution data during multicast distribution is performed, the encryption key and decryption key are changed periodically to improve the confidentiality of the encryption, and it is possible to prevent delays associated with the acquisition (by the client) of new key information (the key and associated information) when the key is changed.”); Peterka, [0082] (“The client will monitor the expiration time for each pay period and request a new set of keys from the caching server.”), [0110] (“All clients on this list [maintained by the server] can then be periodically given new keys using a multicast UDP message which has a new

program segment key encrypted for each participant using that participant's unique key.”).

166. A POSITA would have had reason to implement Peterka's client features in view of Suzuki to acquire and use “key information” about a next key—its “remaining validity time” and “key number”—at a set time *before* the expiration of the current key, and thereby facilitate timely key acquisition and use. Suzuki, [0076]-[0077] (“When the remaining validity time of key A reaches Y (second setting value), the client 51 transmits a key information request 81 to the key management server 31, requesting key information on the next key to be used.”), [0080]-[0081] (“When client 51 receives a multicast packet having the key number of key B from content server 11, it discards the key information regarding key A and stores key B, which was saved as the key to be used next, as the key currently in use.”), [0043]-[0048] (“[I]f the key numbers of both the key currently being used and the key to be used next do not match the key number in the multicast packet, the process of acquiring the key information has not been performed correctly, so an inquiry to that effect is made to the key management server 31 (step A7).”). This would have allowed a POSITA to “eliminat[e] the confirmation delay” that occurs during key changes (Suzuki, [0011]), and realize Peterka's embodiment where “each client keeps track of the keys and their expiration times and actively requests new

keys before the current keys expire so as to avoid service interruptions.” Peterka, [0105].

167. Because Suzuki expressly teaches a specific way to achieve what Peterka expressly discloses (Peterka, [0082], [0105]), a POSITA would have reasonably expected success applying Suzuki’s teachings to Peterka.

2. Claim-by-Claim Analysis

a. Claims 1-2, 10, 12, 16-18, and 20

168. As discussed, Peterka-Suzuki applies Suzuki’s teachings regarding “key numbers” and “remaining validity times” to Peterka’s system. Peterka-Suzuki’s client receives a “key number” and “remaining validity time” that correspond to each received “content encryption key” described above at §§ VIII.A.1.a.ii, and VIII.A.1.a.iii(1) [subsection a)].

169. Peterka-Suzuki uses the “key number” to keep track of keys and the “remaining validity time” to determine when to request a next key. Suzuki, [0043]-[0048] (“If it is determined in step A4 that ‘the two key numbers match,’ the encrypted data in the multicast packet is decrypted using the decryption key of that key number (step A6).”), [0072]-[0075] (“FIG. 5 is a diagram specifically showing the information in each key information management table and the contents of each packet/request/message. In FIG. 5, the key currently in use is indicated as key A, the next key to be used is indicated as key B.... FIG. 6 is a diagram showing an

operation sequence over time when the information shown in FIG. 5 is handled.”), [0076]-[0077] (“When the remaining validity time of key A reaches Y (second setting value), the client 51 transmits a key information request 81 to the key management server 31, requesting key information on the next key to be used. In response to this key information request 81, the key management server 31 transmits a response message 82 having key information relating to the key currently in use (key A) and the next key to be used (key B). As a result, the client 51 has key information relating to the keys A and B, as shown in FIG. 5.”), [0080]-[0081] (“When client 51 receives a multicast packet having the key number of key B from content server 11, it discards the key information regarding key A and stores key B, which was saved as the key to be used next, as the key currently in use.”), FIGs. 2 & 5-6.

170. Peterka-Suzuki’s “key numbers” uniquely identify keys and are *content encryption key identifiers*, as claimed. Peterka-Suzuki’s “remaining validity times”—which correspond to Peterka’s “time remaining in the lifetime of the key” in an ECM or EMM (Peterka, [0121]) and are used to request new keys before the end of a current key’s lifespan—are *content encryption key expiration times*, as claimed.

171. By sending requests for new keys whenever the “remaining validity time” of a key currently in use reaches a particular “setting value,” Peterka-Suzuki

“detect[s]” key rotation boundaries as claimed in Elements [1D] and [16E] (in addition to the key change signaling-based detection I described above in § VIII.A.1.a.v, which included analysis that cited to Suzuki’s teachings and also applies here) and “issu[es]” requests to a license server *ahead of key rotation boundaries* as claimed in Elements [1E] and [16F]. Suzuki, [0033] (“The content server 11, the key management server 31, and each client 5i update the remaining validity time in the key information in the key information management table 21, the key information management table 41, and each key information management table 6i that they manage every time the time passes (performing updates based on their own clock signals, and the like).”), [0036] (“FIG. 4 is a flow chart showing processing relating to key management in the key exchange method in the multicast distribution system according to this embodiment (processing at the point in time when the remaining validity time of the key in use reaches the second setting value).”), [0059] (“When the remaining validity time of the key in use in the key information management table 6i of the client 5i reaches the second setting value, the key management means 51 in the client 5i confirms to that effect (that the remaining validity time value has decreased and reached the second setting value) (step C1), and to obtain key information relating to the next key (decryption key) to be used, a key, information request 81 is sent to the key management server 31 (step C2).”), [0072]-[0075] (“FIG. 5 is a diagram specifically showing the information in

each key information management table and the contents of each packet/request/message. In FIG. 5, the key currently in use is indicated as key A, the next key to be used is indicated as key B.... FIG. 6 is a diagram showing an operation sequence over time when the information shown in FIG. 5 is handled.”), [0076]-[0077] (“When the remaining validity time of key A reaches Y (second setting value), the client 51 transmits a key information request 81 to the key management server 31, requesting key information on the next key to be used. In response to this key information request 81, the key management server 31 transmits a response message 82 having key information relating to the key currently in use (key A) and the next key to be used (key B). As a result, the client 51 has key information relating to the keys A and B, as shown in FIG. 5.”), FIGs. 4-6.

172. For the remaining elements in claims 1 and 16, and for claims 2, 10, 12, 17-18, and 20, Peterka-Suzuki meets each element/claim for the same reasons Peterka alone meets them, as summarized in the table below.

Claim/Element	Section
[1PRE]	§ VIII.A.1.a.i
[1A]	§ VIII.A.1.a.ii
[1B]	<i>See</i> § VIII.B.2.a above
[1C]	§ VIII.A.1.a.iv
[1D]	§ VIII.A.1.a.v; <i>see</i> § VIII.B.2.a above
[1E]	<i>See</i> § VIII.B.2.a above
[1F]	§ VIII.A.1.a.vii
[2]	§ VIII.A.1.b
[10]	§ VIII.A.1.g
[12]	§ VIII.A.1.h

[16PRE]	§ VIII.A.1.i.i
[16A]	§ VIII.A.1.i.ii
[16B]	§ VIII.A.1.a.ii
[16C]	<i>See</i> § VIII.B.2.a above
[16D]	§ VIII.A.1.a.iv
[16E]	§ VIII.A.1.a.v; <i>see</i> § VIII.B.2.a above
[16F]	<i>See</i> § VIII.B.2.a above
[16G]	§ VIII.A.1.a.vii
[17]	§ VIII.A.1.b
[18]	§ VIII.A.1.d
[20]	§ VIII.A.1.h

b. Claims 3-6

173. As discussed above in §§ VIII.B.1 and VIII.B.2.a, in Peterka-Suzuki, Peterka’s client uses Suzuki’s “remaining validity time” as Peterka’s “time remaining in the lifetime of the key” (Peterka, [0121]) for each received content encryption key in an ECM or EMM, and uses Suzuki’s method of “recogni[zing]” that “the remaining validity time value has decreased and reached” a particular “setting value” to determine when to acquire new keys. Suzuki, [0059] (“When the remaining validity time of the key in use in the key information management table 6i of the client 5i reaches the second setting value, the key management means 51 in the client 5i confirms to that effect (that the remaining validity time value has decreased and reached the second setting value) (step C1), and to obtain key information relating to the next key (decryption key) to be used, a key information request 81 is sent to the key management server 31 (step C2).”); *see also* Suzuki, [0033] (“The content server 11, the key management server 31, and each client 5i

update the remaining validity time in the key information in the key information management table 21, the key information management table 41, and each key information management table 6i that they manage every time the time passes (performing updates based on their own clock signals, and the like).”), [0036] (“FIG. 4 is a flow chart showing processing relating to key management in the key exchange method in the multicast distribution system according to this embodiment (processing at the point in time when the remaining validity time of the key in use reaches the second setting value).”), [0072]-[0075] (“FIG. 5 is a diagram specifically showing the information in each key information management table and the contents of each packet/request/message.... [T]he first setting value is indicated by X, and the second setting value is indicated by Y.”), [0076]-[0077] (“When the remaining validity time of key A reaches Y (second setting value), the client 51 transmits a key information request 81 to the key management server 31, requesting key information on the next key to be used.”), FIGs. 4-6.

174. In Peterka-Suzuki, consistent with the teachings of both references, keys are changed periodically. Suzuki, [0003] (“At this time, by periodically changing the key, it is possible to manage clients and further improve the confidentiality of the encryption.”), [0012] (“[T]he feature of the present invention is that when encryption of distribution data during multicast distribution is performed, the encryption key and decryption key are changed periodically to

improve the confidentiality of the encryption, and it is possible to prevent delays associated with the acquisition (by the client) of new key information (the key and associated information) when the key is changed.”); Peterka, [0082] (“The client will monitor the expiration time for each pay period and request a new set of keys from the caching server.”), [0110] (“All clients on this list [maintained by the server] can then be periodically given new keys using a multicast UDP message which has a new program segment key encrypted for each participant using that participant’s unique key.”). Peterka-Suzuki’s use of Suzuki’s “remaining validity time” would have met claim 3 (or would have been obvious to implement in ways that met claim 3) for the same reasons described above at § VIII.A.1.c regarding Peterka alone.

175. Further, Peterka-Suzuki’s client acquires a new key before it is needed (*times the prefetching of a next un-retrieved content encryption key*) when the “remaining validity time” of the current key reaches a fixed “setting value.” *E.g.*, Suzuki, [0076]-[0077]. A POSITA understood that this disclosed prefetching is both *based on an expected expiration of the content encryption key currently being used (claim 4)* and occurring *a fixed duration before the expected expiration of the content encryption key currently being used (claim 5)*, as recited.

176. Moreover, in view of Suzuki’s teachings, a POSITA would have had reason to implement Peterka-Suzuki’s periodic “prefetching”—which occurs a set amount of time before each expiration of a series of periodically-changing keys—to

occur at no later than *at a duration calculated as a fixed percentage of a fixed periodic content encryption key expiration interval (claim 6)*. As noted above, Suzuki expressly teaches a client calculating the “setting value” at which it requests a new key as a “certain percentage of the time from the initial value” of the “remaining validity time of the key.” Suzuki, [0070].

C. Ground 3: Peterka in View of Bocharov Renders Claims 1-6, 10, 12, 14, 16-18, and 20 Obvious

177. To the extent retrieving manifest files from a content delivery server is not considered disclosed or rendered obvious by Peterka alone, modifying Peterka in view of Bocharov would have rendered obvious claims 1-6, 10, 12, 14, 16-18, and 20, as discussed below.

1. Peterka-Bocharov

178. As discussed above in § VII.A, Peterka describes techniques for distributing streaming media content to client devices over a network. In Peterka, servers are configured to stream encrypted media content to client devices, which apply decryption keys to playback content. Peterka, [0002] (“[T]he invention relates to providing a preview portion of a program distributed to clients on the network.”), [0004] (“With the advent of multicasting networks, program content such as movies and music can now be distributed in multicast transmissions across networks”), [0011] (“[O]ne embodiment may allow the user to purchase the program content as well as provide the user with a key that is operable to decrypt the encrypted portion

of the program.”), [0038] (“Referring to FIG. 1, a block diagram of a content distribution system 100 is shown.”), [0078]-[0082] (“For a quasi-PBT approach, a caching server may divide the content into pay segments and assign segment keys to them.... When the client negotiates encryption keys with the caching server, it will start receiving the multicast content.”), [0091]-[0093] (“As can be seen in FIG. 7A an initial free preview period provides for client computers to request an initial key for viewing an encrypted portion of the content.... Content key 0 is provided to the user in one embodiment for obtaining the free preview.”), [0127] (“One method of implementing an initial viewing period is to distribute a common key to potential participants ahead of time, e.g., prior to distribution of program content.... [T]he clients can decrypt the content key utilizing the group key and then utilize the content key to decrypt the encrypted program content.”), FIG. 7A.

179. Peterka also teaches streaming content at different “quality (i.e., bit rates)” which the client can “purchase” and “adjust... in accordance with user selection.” Peterka, [0054].

180. As discussed above at § VII.C, Bocharov discloses techniques that allow a client to obtain fragments of media content “in whichever encoding the client chooses” using “manifest files” that are provided with the media fragments. Bocharov, [0049]; *see also* Bocharov, [0007] (“[T]he server provides uniform media fragment responses to media fragment requests, thereby allowing existing Internet

cache infrastructure to cache streaming media data.”), [0014] (similar), [0016] (“Each fragment contains metadata and media content.”), [0020]-[0023] (“The encoder interface component 120 may specify a particular format for received encoded media data, such as an MP4 or other media container (e.g., MKV).... Encoders may place metadata information in the boxes such as the type of encoding used to encode the encoded media data, as well as the encoded media data itself.”), [0049]-[0062] (“The client manifest includes a static portion that is a union of each of the encoder manifests that describes the available encodings, and a dynamic portion that describes media fragments received by the server from the encoders thus far.”), FIG. 5. Bocharov expressly teaches issuing clients manifest files that specify different encodings available for streaming media fragments. *E.g.*, Bocharov, [0007] (“Each fragment contains metadata information that describes the encodings available on the server and the encoding of the fragment in addition to the media content of the fragment for playback by the client.”).

181. Both references stream similar types of media (*e.g.*, MPEG 4 files) and apply segmentation to divide the media into fragments for distribution. Peterka, [0080] (“For a quasi-PBT approach, a caching server may divide the content into pay segments and assign segment keys to them.”), [0119] (discussing MPEG 4 payload); Bocharov, [0020] (“The encoder interface component 120 may specify a

particular format for received encoded media data, such as MP4⁸ or other media container (e.g., MKV).”), [0034] (“The fragment_count specifies the number of immediate subsequent fragments of the same track that this box is referencing. The fragments are listed in the same order as they appear in the MP4 stream.”), [0038]-[0039] (“Whether using MP4 or other container formats, the encoder may add additional boxes or information to the stream, such as a ‘ManifestBox’ that contains a manifest describing the media element.”).

182. In view of Bocharov, a POSITA would have been motivated to implement Peterka’s client device to issue a client manifest request to a server to obtain manifests (Bocharov, [0053], FIG. 5)—transmitted *e.g.*, in the media fragments that also contain the media content (Bocharov, [0023]-[0024])—which (1) included information allowing a client to request delivery of streaming media content, such as information indicative of a URL for the media content (Bocharov, [0030]-[0031]); (2) identified different “quality” or “bit rate” encodings that the client could request or switch to (Bocharov, [0030]-[0031], [0048]-[0050], [0060]); and (3) provided a list of fragments in the media stream with information indicative

⁸ Video content within an MP4 file is encoded with a standard called MPEG-4. For evidence corroborating this knowledge of a POSITA, *see, e.g.*, <https://www.cloudflare.com/learning/video/what-is-mp4/>.

of a URL for each fragment, so that individual clients could reference the manifests in the media stream to request timely delivery of fragments at a desired encoding and maintain a continuous stream (*id.*).

183. Implementing Peterka’s system to use Bocharov’s manifests would have provided an effective solution for enabling clients to timely request delivery of streaming media content—*e.g.*, at a particular encoding/quality/bit rate—and maintain a continuous stream. Such an implementation using Bocharov’s manifests would have met claims 1-6, 10, 12, 14, 16-18, and 20 for the reasons I discuss below in §§ VIII.C.2.a and VIII.C.2.c.

184. Implementing Peterka’s system to use Bocharov’s manifests would also have provided an effective solution for allowing the content server to adjust streaming bit rates based on the state of the network and clients to select from (and switch between) multiple versions of a media program having different encodings/qualities/bit rates. Peterka acknowledged the benefit of providing different qualities of streams that could accommodate changing user demands and network conditions, and Bocharov’s manifests would have allowed Peterka’s system to embed, within media segments provided to clients, both media content and information needed to switch between different encodings/qualities/bit rates of the media stream. Peterka, [0054] (“Content may be offered at a different quality (*i.e.*, bit rates), either as separate streams or as layered streams with each additional layer

adding quality to the content. The client may report the highest bit rate it can consume and be offered to purchase the content at that quality or less. The server may also adjust the rate in real time based on the immediate state of the network.... The client may also adjust the bit rate perceived by the user, in accordance with user selection.”); Bocharov, [0049] (“The client manifest includes a static portion that is a union of each of the encoder manifests that describes the available encodings, and a dynamic portion that describes media fragments received by the server from the encoders thus far.”), [0053] (“The origin server 510 receives a manifest request 530 from a client 515. The origin server 510 generates 535 a client manifest based on the latest media fragment information. The origin server 510 provides a client manifest response 540 to the client 515.”). Indeed, Bocharov itself expressly describes the use of manifests to improve tiered-quality purchasing models similar to Peterka’s (Peterka, [0054])—*e.g.*, when explaining that a “system may enable or disable switching to particular bit rates based on whether the conditions of the payment model have been met” by including needed “information” “in the manifest provided to the client.” Bocharov, [0060]. Such an implementation—in which Bocharov’s manifests are used to enable switching between encodings/qualities/bit rates—would additionally have met claims 12 and 20, for the reasons I discuss below in § VIII.C.2.b.

2. Claim-by-Claim Analysis

a. Claims 1-6, 10, 12, 16-18, and 20

185. Elements [1C] and [16D] recite “[retrieve/retrieving] content and manifest files from a content delivery server.” For the reasons discussed above at § VIII.C.1, Peterka-Bocharov applies Bocharov’s manifest file teachings to Peterka’s system, such that Peterka-Bocharov’s clients retrieve the manifests taught in Bocharov as part of the media stream, and the manifests include encoding information that the client may request or switch to.

186. As discussed above at § VIII.A.1.a.iv, Peterka discloses a client (*e.g.*, client computer 112) that *retrieves content from a content delivery server* (*e.g.*, origin 108, a context exchange 116, or caching server). Peterka, [0038]-[0039] (“Referring to FIG. 1, a block diagram of a content distribution system 100 is shown.”), [0040] (“Content objects of an external original processor 118 can be preloaded to a content exchange(s) allocated to provide those content objects.”), [0060] (“Origin Content Server”), [0071] (same), [0083] (“In block 304 encrypted material is provided for distribution to a client.”), [0135] (“The origin content server is intended to illustrate a server which stores or controls access to program content.”), FIG. 1.

187. This is consistent with Bocharov’s teachings of clients like Peterka’s which likewise retrieve media content from a content delivery server by requesting

media “fragments.” Bocharov, [0006]-[0007] (“A smooth streaming system is described herein that provides a stateless protocol between the client and server in which the server embeds incremental information in media fragments that eliminates the usage of a typical control channel.”), [0015] (“The client requests fragments of media from a server that include a portion of the media.”), [0021]-[0022] (“The fragment data store 140 stores received media fragments and the created index table of fragments to provide to clients based on received client requests.”). For further context, see my discussion above in § VII.C.

188. Bocharov expressly discloses *retrieving manifest files from a content delivery server*. Bocharov, [0053] (“The origin server 510 receives a manifest request 530 from a client 515” which “provides a client manifest response 540 to the client 515.”), FIG. 5; *see also* Bocharov, [0023]-[0025] (“The client interface component 150 receives client requests for media fragments and provides manifest data and media fragments to clients.”), [0030]-[0035] (“The client manifest lists the decoding information as well as information for all the fragments that the server has archived so far.”), [0049]-[0052] (“[T]he system builds a manifest to satisfy the client request based on the fragments received and the encoder information received when the system initially requested encoder manifests.”). For further context, see my discussion above in § VII.C.

189. Bocharov also expressly teaches retrieving multiple manifest files, *e.g.*, updated manifests with information that may not have been available in an initial manifest that are transmitted with subsequent media fragments. Bocharov, [0024] (“subsequent manifests provided with each media fragment”), [0032] (“the server may send the fragment along with manifest information about the next few fragments”), [0023]-[0025] (“By combining the initially received manifest with subsequent manifests provided with each media fragment, the client can build an up to date manifest that includes complete information about the media event from the start up until the current time.”). Thus, the client in Peterka-Bocharov *retrieves content and manifest files from a content delivery server*, as claimed in Elements [1C] and [16D].

190. For the remaining elements in claims 1 and 16, and for claims 2-6, 10, 12, 17-18, and 20, Peterka-Bocharov meets each element/claim for the same reasons Peterka alone meets them, as summarized in the table below.

Claim/Element	Section
[1PRE]	§ VIII.A.1.a.i
[1A]	§ VIII.A.1.a.ii
[1B]	§ VIII.A.1.a.iii
[1C]	<i>See § VIII.C.2.a above</i>
[1D]	§ VIII.A.1.a.v
[1E]	§ VIII.A.1.a.vi
[1F]	§ VIII.A.1.a.vii
[2]	§ VIII.A.1.b
[3]	§ VIII.A.1.c
[4]	§ VIII.A.1.d

[5]	§ VIII.A.1.e
[6]	§ VIII.A.1.f
[10]	§ VIII.A.1.g
[12]	§ VIII.A.1.h
[16PRE]	§ VIII.A.1.i.i
[16A]	§ VIII.A.1.i.ii
[16B]	§ VIII.A.1.a.ii
[16C]	§ VIII.A.1.a.iii
[16D]	<i>See § VIII.C.2.a above</i>
[16E]	§ VIII.A.1.a.v
[16F]	§ VIII.A.1.a.vi
[16G]	§ VIII.A.1.a.vii
[17]	§ VIII.A.1.b
[18]	§ VIII.A.1.d
[20]	§ VIII.A.1.h

b. Claims 12 and 20: “...wherein actual content encryption key rotation boundaries are detected based on real-time in-band notifications.”

191. Claims 1 and 16 recite detecting “content encryption boundaries between periods of use of different content encryption keys in decrypting retrieved content”—while claims 12 and 20 recite detecting “actual content encryption key rotation boundaries.” Based on the plain language of the claims, a POSITA would have understood claims 12 and 20 to at minimum encompass a distinct or separate detection of “actual boundaries” that occurs in addition to the detection recited in claims 1 and 16.

192. Peterka-Bocharov additionally and alternatively meets claims 12 and 20 because, in addition to the boundary detection discussed above at § VIII.A.1.a.v, Peterka-Bocharov’s client also detects content encryption key boundaries based on

information in Bocharov's manifest files that tell the client to switch to a different encoding of a media stream, as discussed below.

193. As discussed above in § VIII.C.1, Peterka-Bocharov's client manifest can be used to cause a client to switch different encoding (and different quality) of the media stream due to, *e.g.*, failure to meet payment requirements. Bocharov, [0060] (“[T]he system may enable or disable switching to particular bit rates based on whether the conditions of the payment model have been met.... This information may be included in the manifest provided to the client.”), [0048]-[0050] (“Based on the manifest, the client can begin requesting fragments in whichever encoding the client chooses. For example, the client may initially select a low bit rate encoding and select higher bit rate encodings for subsequent fragments until network bandwidth limits the client's ability to receive the fragments at a bit rate.”).

194. A POSITA would have understood that such a use of Peterka-Bocharov's manifests constituted *real-time in-band notifications*, as claimed, because Bocharov's manifests are transmitted as part of the media content stream along with corresponding media content (*in-band*)—Bocharov, [0007] (“server embeds incremental information in media fragments that eliminates the usage of a typical control channel”); '178 Patent, 4:13-15 (“The packager is responsible for providing in-band notification to the client for the key change.”)—and the client would have to switch to a different quality encoding of the stream upon receiving

such a notification (in *real-time*). See also Bocharov, [0016] (“Each fragment contains metadata and media content. The metadata may describe useful information about the media content, such as the bit rate at which the media content was encoded, where the media content fits into a larger media element.... The client uses this information to place the fragment into a storyboard of the larger media element and to properly decode and playback the media content.”), [0025] (“This allows the system 100 to include manifest information about the subsequent fragments in the current fragment response.... Thus, in response to any particular fragment request, build manifest component 160 provides manifest information about as many subsequent fragments as the system 100 has received up to that point.”), [0032] (“Then, the server may send the fragment along with manifest information about the next few fragments. The client can use this information to know what is coming and adapt appropriately.... Thus, the client can adapt using the advance metadata as a hint.”).

195. It was conventional for different encodings of a media stream (Peterka, [0054]; Bocharov, [0049], [0060]) to be encrypted using different keys. For documentary evidence corroborating this background knowledge of a POSITA, see, for example, the Yuan (EX-1023) reference. Yuan, pages 517-519 (“To support access to a video sequence at different quality levels in non-scalable coding, the same video would have to be compressed into separate files of different qualities,

with each file encrypted with a different encryption key.”).

196. Thus, a POSITA would have understood that a notification in a manifest instructing a client to switch to a different encoding for a stream would have satisfied claim 12’s requirement of actual *key rotation boundaries*, because a POSITA understood that switching to a different encoding for a stream entailed needing a new key (*e.g.*, a key specific to the switched-to fragment) to decrypt content, and therefore corresponded to an actual key rotation boundary. For documentary evidence corroborating a POSITA’s background knowledge regarding the use of manifests to convey information about encryption, see, for example, the Pestoni and Lewis references. Pestoni, 8:39-63 (known to obtain a manifest in advance to secure decryption information); Lewis, 7:38-45 (known to use a manifest file to manage keys dealing with time windows and rental periods for “enhanced security.”). The ’178 Patent specification itself acknowledges existing “protocols” in 2011 that “employ[ed] segment-based rate adaptation to deal with fluctuations in bandwidth, whereby segment boundaries provide natural demarcation points for switching bitrates” (’178 Patent, 2:31-37), and I understand that the proposition that actual key rotation boundaries occur when a client switches between encodings/qualities/bit-rates is consistent with Patent Owner’s broad infringement arguments in the 1376 ITC Investigation.

197. To the extent the above-described notification is considered insufficient

to meet claims 12 and 20, it would have been obvious to implement Peterka-Bocharov's manifest to provide information needed for clients to detect the need for and acquire the keys necessary to switch to a different encoding. Such an implementation would have been consistent with the "pull method" for key distribution used in Peterka-Bocharov, in which clients actively request new keys, and accordingly a POSITA would have reasonably expected success with such an implementation. Peterka, [0053] (Pay-By-Time "PBT"), [0078]-[0082] (discussing PBT, "The client will monitor the expiration time for each pay period and request a new set of keys from the caching server."), [0105]-[0106] ("Under the pull model, each client keeps track of the keys and their expiration times and actively requests new keys before the current keys expire so as to avoid service interruptions."), [0110] ("Under a pull key distribution model, each client requests a new key and is individually responded to by the key distribution server.").

- c. **Claim 14: "The method of claim 12, wherein content encryption key rotation boundaries are detected based on notifications present in the content manifest file, the content manifest file being selected from an m3u8 file, an IIS Smooth Streaming manifest file, a DASH MPD file and a proprietary manifest file."**

198. A POSITA would have understood that claim 14 encompasses an interpretation where the term "content manifest file" corresponds to one of the "manifest files" recited in independent claim 1, from which claim 14 depends. Claim 14 requires the recited content manifest file to be "*selected from*" a limited list of

alternative manifest file types. I understand that only one listed alternative must be satisfied to meet the claim. All of the alternatives would have been obvious, including an IIS Smooth Streaming manifest file, as discussed below.

199. Bocharov teaches that the manifest file described above in § VIII.C.2.b can be an Internet Information Server (“IIS”) smooth streaming manifest file. Bocharov, [0036] (“FIG. 2 is a block diagram that illustrates an operating environment of the smooth streaming system using Microsoft Windows and Microsoft Internet Information Server (IIS), in one embodiment.”), [0038] (“For example, using IIS...”), [0010] (“FIG. 2 is a block diagram that illustrates an operating environment of the smooth streaming system using Microsoft Windows and Microsoft Internet Information Server (IIS), in one embodiment.”), FIG. 2; *see also* Bocharov, Abstract (“smooth streaming system”), [0014] (same), [0030] (describing a “typical client manifest” that includes “SmoothStreamingMedia” fields).

200. A POSITA would have had reasons to apply Bocharov’s use of an IIS Smooth Streaming manifest file in Peterka-Bocharov since Bocharov confirms that such a manifest file was suitable to carry fragment information of the types disclosed in Bocharov, for example, information about encoding qualities. Bocharov, [0030]. Further, a POSITA would have recognized benefits of using a known manifest file format that could be parsed by appropriate software at client devices according to

the dictates of the IIS format and the teachings of Peterka and Bocharov. This would have been a simpler and more convenient alternative to using, *e.g.*, a new manifest file format that was not defined in a standard. *Id.*

D. Ground 4: Peterka in View of Suzuki and Bocharov Renders Claims 1-6, 10, 12, 14, 16-18, and 20 Obvious

201. To the extent retrieving manifest files from a content delivery server is not considered rendered obvious by the Peterka-Suzuki combination (as I discussed above in § VIII.B), modifying Peterka-Suzuki in view of Bocharov—for the same reasons a POSITA would have modified Peterka alone in view of Bocharov (which I discussed above in § VIII.C.1)—would also separately have rendered obvious claims 1-6, 10, 12, 14, 16-18, and 20.

1. Peterka-Suzuki-Bocharov

202. As I discussed above in § VIII.B, a POSITA had reasons (and would have been motivated) to implement Peterka’s client in view of Suzuki to acquire and use “key information” about a next key—the key’s “remaining validity time” and “key number”—at a set time *before* the expiration of the current key, thereby facilitating timely key acquisition and use. Suzuki, [0076]-[0077], [0080]-[0081], [0043]-[0048].

203. For the same reasons discussed I discussed above in § VIII.C.1, in view of Bocharov’s teachings, a POSITA would have been motivated to further modify Peterka-Suzuki’s client to issue a client manifest request to a server to obtain

manifests that include information that enable the client to request media and switch between different quality encodings. Bocharov, [0049]; *see also* Bocharov, [0007], [0014], [0016], [0020]-[0023], [0049]-[0062], FIG. 5. For additional context, see my discussion above regarding Bocharov in § VII.C.

204. The same reasoning described for Peterka alone applies to Peterka-Suzuki because the content manifest teachings of Bocharov are complementary to and do not conflict with the key identifier and key acquisition teachings of Suzuki. A POSITA would also have reasonably expected success with an implementation combining Peterka-Suzuki and Bocharov for the same reasons described *supra* §§ VIII.B.1 and VIII.C.1, for each individual combination.

2. Claim-by-Claim Analysis

a. Claims 1-6, 10, 12, 14, 16-18, and 20

205. As discussed, Peterka-Suzuki-Bocharov applies (to Peterka's system) Suzuki's teachings regarding key numbers and remaining validity times and Bocharov's teachings regarding manifest files. In such an implementation, a client receives a "key number" and "remaining validity time" that correspond to each received "content encryption key" described above in §§ VIII.A.1.a.ii, VIII.A.1.a.iii(1) [subsection a)], and uses the "key number" taught by Suzuki to keep track of keys and the "remaining validity time" to determine when to request a new

key. Suzuki, [0043]-[0048], [0072]-[0077], [0080]-[0081], FIGs. 2 & 5-6. For additional context, see my discussion above in § VIII.B.2.a.

206. The client separately retrieves the manifests taught in Bocharov as part of the media stream, and the manifests include information regarding available stream encodings that the client may request or switch to. Bocharov, [0023]-[0025], [0030]-[0035], [0049]-[0053], FIG. 5. For additional context, see my discussion above in § VIII.C.2.a.

207. Peterka-Suzuki-Bocharov meets each element/claim for the same reasons described in the sections summarized in the table shown below.

Claim/Element	Section
[1PRE]	§ VIII.A.1.a.i
[1A]	§ VIII.A.1.a.ii
[1B]	§ VIII.B.2.a
[1C]	§ VIII.C.2.a
[1D]	§§ VIII.A.1.a.v, VIII.B.2.a
[1E]	§ VIII.B.2.a
[1F]	§ VIII.A.1.a.vii
[2]	§ VIII.A.1.b
[3]	§§ VIII.A.1.c, VIII.B.2.b
[4]	§§ VIII.A.1.d, VIII.B.2.b
[5]	§§ VIII.A.1.e, VIII.B.2.b
[6]	§§ VIII.A.1.f, VIII.B.2.b
[10]	§ VIII.A.1.g
[12]	§§ VIII.A.1.h, VIII.C.2.b
[14]	§ VIII.C.2.c
[16PRE]	§ VIII.A.1.i.i
[16A]	§ VIII.A.1.i.ii
[16B]	§ VIII.A.1.a.ii
[16C]	§ VIII.B.2.a
[16D]	§ VIII.C.2.a

[16E]	§§ VIII.A.1.a.v, VIII.B.2.a
[16F]	§ VIII.B.2.a
[16G]	§ VIII.A.1.a.vii
[17]	§ VIII.A.1.b
[18]	§§ VIII.A.1.d, VIII.B.2.b
[20]	§§ VIII.A.1.h, VIII.C.2.b

E. Ground 5: Peterka in View of Bocharov, Peterka308, and Chen Additionally Renders Claims 7-8 and 19 Obvious

208. For at least the reasons I explain below, implementing Peterka-Bocharov in view of Peterka308 and Chen would have rendered obvious claims 7-8 and 19, which recite specific ways in which content encryption key identifiers are calculated.

1. Peterka-Bocharov-Peterka308-Chen

209. Peterka teaches a “signaling method... to indicate the implementation of a new key,” suggesting a client acquiring keys for multiple future media segments and selecting the correct key from among multiple keys in its possession using *multi-bit* signals embedded in streaming media packets. Peterka, [0117] (“*if* a single bit is used”), [0118]-[0121] (“[T]he extended portion of the header could include at least the content key parity bit to indicate key changes.”), FIG. 14. At minimum, it would have been obvious to implement Peterka’s client to acquire keys for multiple future segments to help reduce the “load” of client requests at segment boundaries, and a POSITA would have reasonably expected success doing so. *See* Peterka, [0080] (noting that a pull model for key distribution, in which “[a]ll consumer computers...

negotiate a key for each segment” may “generate[] a large load on the... server on the segment boundaries”). The POSITA would have appreciated, for example, that distributing keys for additional segments beyond just the current and next segments would have beneficially reduced the load of client requests that needed immediate service in order to avoid service interruptions for Peterka’s clients. Peterka, [0105] (“Under the pull model, each client keeps track of the keys and their expiration times and actively requests new keys before the current keys expire so as to avoid service interruptions.”). For corroboration of this background knowledge of a POSITA, see, for example, the Pestoni reference (EX-1012), at 12:1-46 (describing “key box” with keys for multiple future segments, and noting: “In some embodiments that utilize the key box techniques..., the overall ratio of license requests to content decryption keys may be reduced relative to the embodiment of FIG. 2... for a given quantity of content decryption keys utilized for content, the embodiment of FIG. 4 may require license server(s) 120 to process fewer content license requests.”).

210. As discussed above in § VII.D, Peterka³⁰⁸ discloses techniques for correlating content encryption keys and key identifiers with information indicating a location in a media stream where the content encryption key should be applied. Specifically, Peterka³⁰⁸ proposes using a “time stamp” to derive the appropriate content encryption key for a “segment” of encrypted media content, where the “time stamp” indicates the “relative time value” from the beginning of the media content to

the start of the segment. Peterka308, [0025] (“[A] time stamp, which indicates the relative time value from the beginning of the recording to the start of the segment, is generated and stored at the time that each unique CEK is determined.”).

211. In the context of a system like Peterka’s, which distributes keys to client devices in advance of streaming segments that require use of such keys, a POSITA would have sought to apply Peterka308’s suggested use of segment locators (*e.g.*, time stamps) as content encryption key identifiers to provide a more robust method (an absolute point in time) of signaling to client devices which content encryption key (*e.g.*, which CK) to apply at the start of each segment, and would have reasonably expected success doing so. Such an implementation would be beneficial, for example, for a client that held keys for multiple future segments, because it could readily compare the current playback location of a media stream to the respective location information of the future segment keys to determine which key to apply at the next key rotation boundary. *See* Peterka, [0117]-[0121] (“[A] client can check the predetermined bit in a packet and determine the proper key use.”); Peterka308, [0025] (“During playback, the master key and the time stamp for a particular segment may be utilized, at least in part, to derive the CEK for that segment so that the encrypted content for that segment stored in the hard drive 202 or other media storage may be decrypted.”). Implementing Peterka308’s teaching

would have beneficially allowed clients to more effectively manage keys obtained for segments multiple periods in advance.

212. Further, to the extent Peterka does not expressly disclose what criteria content servers would apply to subdivide media content into cryptographic “segments” (where each segment is encrypted by a different key), Chen discloses criteria that were known for this task. For additional context, see my discussion above in § VII.E. For example, Chen teaches a first option (“fixed crypto period”) involving defining “a crypto time interval and chang[ing] the key each time the crypto time-interval passes.” Chen, [0034] (“Fixed crypto period: Define a crypto time interval and change the key each time the crypto time-interval passes.”), [0037]-[0038] (“Figure 2A illustrates this fixed crypto period encoding scheme 200a, showing an encoded/encrypted MPEG-II stream 202, divided into a plurality of segments 204.... Each segment 204 is encrypted according to a different encryption key.”), FIG. 2A.

213. A second option (“fixed number of packets”) that Chen teaches involves determining “a number of content packets ‘n’ corresponding to a suitable time interval and chang[ing] the cryptographic key every ‘n’ packets.” Chen, [0034] (“Fixed number of packets: Determine a number of content packets ‘n’ corresponding to a suitable time interval and change the cryptographic key every ‘n’ packets.”), [0039]-[0041] (“The ‘fixed number of packets’ scheme then examines

the encoding of the MPEG-II stream for an item of program content to determine a suitable number 'n' of MPEG-II packets which correspond to a crypto period.... Each segment 204 is encrypted according to a different encryption key.”), FIG. 2B (illustrating an encoding scheme “200b” that uses a fixed number of MPEG-II packets per crypto interval).

214. A third option (“fixed ‘marker’ count”) that Chen teaches involves using “a suitable MPEG-II field type as a ‘marker,’ such as an I-frame header,” and “chang[ing] the cryptographic key every time ‘n’ markers have passed in the stream.” Chen, [0034] (“Fixed ‘marker’ count: Using a suitable MPEG-II field type as a ‘marker’, such as an I-frame header, change the cryptographic key every time ‘n’ markers have passed in the stream, where ‘n’ is selected to produce a suitable crypto period.”), [0042]-[0044] (“In this scheme, using a suitable, recurring element of MPEG-II encoding as a stream segment delimiter (marker), a number ‘n’ of stream segments is determined that correspond to a suitable crypto period. The stream is then encrypted in ‘crypto segments’ defined by ‘n’ markers.... Each crypto segment is encrypted using a different cryptographic key.”), FIG. 2C (illustrating an encoding scheme “200c” that uses MPEG-II “markers” to delimit crypto intervals in the MPEG-II stream).

215. A POSITA would have recognized each of these options as a viable design alternative that was obvious to try in view of the limited number of methods

available to specify locations in a media stream. Each of Chen’s options also carried the obvious benefit of allowing the encoding device and decoding devices to reference a common metric that could be readily obtained and used to select an appropriate key at each boundary, at the appropriate time, corresponding to the same location in the media content at which the encoding device rotated encryption keys. Clients would thus be afforded an additional basis for determining and predicting upcoming key rotations, thus improving the reliability of the system in the event of errors such as corrupted bits in RTP packets. Moreover, when distributing content encryption key identifiers that reference locations of segment boundaries according to the teachings of Peterka and Peterka308, it would have been obvious for the identifiers to relate to the corresponding segmentation method applied based on the teachings of Chen—*e.g.*, identifiers that indicate the elapsed time of the media stream according to a crypto time interval, a number of content packets, or a number of markers (*e.g.*, I-frames) passed in the stream.

2. Claim-by-Claim Analysis

- a. **Claims 7 and 19: “The [method of claim 1 / computerized device of claim 16] wherein subsequent content encryption key identifiers are predictable based on a predetermined known progression.”**

216. Peterka308 discloses content encryption key identifiers in the form of “time stamps,” which a POSITA would have understood were *predictable based on a predetermined known progression* (*e.g.*, every “five minutes has elapsed”).

Peterka308, [0024] (“[A] timer may indicate to the DRM component 204 when five minutes has elapsed, or is about to elapse, so that the DRM component 204 may generate a second CEK to encrypt the second segment 304 and store the encrypted second segment 304 in the hard drive 202 or other media storage.”), [0029] (“In one embodiment, sequential playback of the content is effectuated by DRM component 204 remembering the last CCI [(copy control information)] element utilized.”), [0032] (“The set of content is characterized as a plurality of segments that each has a segment length according to a predetermined time interval.”).

217. Chen further discloses techniques for segmenting media content based on predetermined known progressions such as fixed time intervals, fixed number of packets in a content stream, and fixed “marker” count (e.g., fixed number of I-frames passed in an MPEG-II stream). Chen, [0034]-[0044] (describing three options for subdividing media content into cryptographic segments: (1) “fixed crypto period,” (2) “fixed number of packets,” and (3) “fixed ‘marker’ count”), FIGs. 2A-2C. For additional context, see my discussion above in § VII.E.

218. For the reasons described above in § VIII.E.1, it would have been obvious to combine Peterka, Bocharov, Peterka308, and Chen such that subsequent content encryption key identifiers would indicate locations in a media stream that the corresponding content encryption key should apply based on a time stamp, time interval, packet count, or marker count—each of which is *predictable based on a*

predetermined known progression as claimed, at least to the same extent this limitation is described in the '178 Patent, which describes assigning identifiers “based off” of known progressions such as “sequential integer value[s]” and “wall clock time offset[s].” Cf. '178 Patent, 5:63-6:1 (“In one embodiment, the content encryption key identifiers are based off a next sequential integer value, offset from the previous content encryption key identifier. In another embodiment, the content encryption key identifiers are based off the wall clock time offset from the time the initial content encryption key identifier was generated.”).

- b. Claim 8: “The method of claim 7, further comprising: the identifier being calculated as monotonically increasing sequential integer values based on the number of segments or video frames generated during a fixed periodic content encryption key expiration interval.”**

219. Chen discloses techniques for segmenting media content based on criteria such as a count of frame “markers” (*e.g.*, I-frames) that pass in a video stream. Chen, [0034]-[0044] (“Fixed ‘marker’ count: Using a suitable MPEG-II field type as a ‘marker’, such as an I-frame header, change the cryptographic key every time ‘n’ markers have passed in the stream, where ‘n’ is selected to produce a suitable crypto period.”), FIGs. 2A-2C (illustrating an encoding scheme “200c” that uses MPEG-II “markers” to delimit crypto intervals in the MPEG-II stream). For additional context, see my discussion above in § VII.E.

220. As described above (§ VIII.E.1), a POSITA had reason to implement Peterka-Bocharov-Peterka308-Chen such that content encryption key identifiers indicate locations in a media stream that a content encryption key should be applied to ***based on*** a count of frame “markers” passed in the media stream.

221. A POSITA understood the frame marker count provides an identifier that is calculated as monotonically increasing sequential integer values (because frame number is an integer that increases over time) and based on a number of video frames generated during a fixed periodic content encryption key expiration interval (*e.g.*, corresponding to the “fixed” marker count of Chen).

IX. CONCLUSION

222. As explained above, the Challenged Claims of the '178 Petition are rendered obvious by the combinations discussed above the Challenged Claims are unpatentable under 35 U.S.C. § 103.

223. I reserve the right to supplement my opinions in the future to respond to any arguments that Patent Owner or its expert(s) may raise and to take into account new information as it becomes available to me.

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: 09/30/2024



Anthony J. Wechselberger