

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of: Kevin J. Ma et al.
U.S. Patent No.: 9,313,178 Attorney Docket No.: 50095-0082IP1
Issue Date: April 12, 2016
Appl. Serial No.: 14/266,368
Filing Date: April 30, 2014
Title: METHOD AND SYSTEM FOR SECURE OVER-THE-TOP LIVE
VIDEO DELIVERY

Mail Stop Patent Board

Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

**PETITION FOR *INTER PARTES* REVIEW OF UNITED STATES PATENT
NO. 9,313,178 PURSUANT TO 35 U.S.C. §§ 311–319, 37 C.F.R. § 42**

TABLE OF CONTENTS

I.	REQUIREMENTS FOR IPR UNDER 37 C.F.R. § 42.104.....	1
A.	Standing	1
B.	Challenge and Relief Requested	1
II.	SUMMARY OF THE '178 PATENT	3
A.	Brief Description.....	3
B.	Summary of the Prosecution History.....	4
C.	Level of Ordinary Skill in the Art.....	5
III.	CLAIM CONSTRUCTION UNDER 37 C.F.R. §§ 42.104(b)(3)	5
IV.	THE CHALLENGED CLAIMS ARE UNPATENTABLE.....	6
A.	[Ground 1A]: Peterka in view of Bocharov.....	6
1.	Peterka	6
2.	Bocharov	10
3.	Obviousness Based on Peterka-Bocharov Combination.....	12
B.	[Ground 1B]: Peterka in view of Bocharov, Peterka308, and Chen	45
1.	Peterka308	45
2.	Chen.....	46
3.	Obviousness Based on Peterka-Bocharov-Peterka308-Chen Combination	46
C.	[Ground 1C]: Peterka in view of Bocharov and Balraj	58
1.	Balraj	58
2.	Obviousness Based on Peterka-Bocharov-Balraj Combination ...	59
D.	[Ground 1D]: Peterka in view of Bocharov and Kelly	61
1.	Kelly	61
2.	Obviousness Based on Peterka-Bocharov-Kelly Combination ...	61
E.	[Ground 1E]: Peterka in view of Bocharov, Kelly, and Eisen	64
1.	Eisen	64
2.	Obviousness Based on Peterka-Bocharov-Kelly-Eisen Combination	64
V.	MANDATORY NOTICES UNDER 37 C.F.R. §42.8.....	66
A.	Real Parties-In-Interest Under 37 C.F.R. §42.8(b)(1)	66
B.	Related Matters Under 37 C.F.R. §42.8(b)(2)	66
C.	Lead And Back-Up Counsel Under 37 C.F.R. §42.8(b)(3).....	66
D.	Service Information	67
VI.	PAYMENT OF FEES – 37 C.F.R. §42.103	67
VII.	CONCLUSION.....	67

EXHIBITS

APPLE-1001	U.S. Patent 9,313,178 to Ma et al. (“the ’178 patent”)
APPLE-1002	Prosecution History of the ’178 Patent (Serial No. 14/266,368)
APPLE-1003	Expert Declaration of Dr. Aviel Rubin, Ph.D.
APPLE-1004	U.S. Pub. 2002/0172368 (“Peterka”)
APPLE-1005	U.S. Pub. 2010/0235528 (“Bocharov”)
APPLE-1006	European Patent Pub. 1 418 756 A2 (“Chen”)
APPLE-1007	U.S. Pub. 2009/0254708 (“Balraj”)
APPLE-1008	U.S. Pub. 2012/0254456 (“Visharam”)
APPLE-1009	U.S. Pub. 2008/0270308 (“Peterka308”)
APPLE-1010	U.S. Pub. 2005/0138362 (“Kelly”)
APPLE-1011	U.S. Pub. 2011/0067012 (“Eisen”)
APPLE-1012	U.S. Pub. 2011/0099594 (“Chen594”)
APPLE-1013	Chow et al., <i>A White-Box DES Implementation for DRM Applications</i> , Pre-Proceedings for ACM DRM-2002 Workshop (October 15, 2002)
APPLE-1014	RFC793: <i>Transmission Control Protocol</i> (September 1981)
APPLE-1015	RFC2616: <i>Hypertext Transfer Protocol – HTTP/1.1</i> (June 1999)

CLAIM LISTING

Element	Claim Language
[1.P]	A method for handling secure distribution of content comprising:
[1.1]	initiating a media playback request and receiving a playback request response;
[1.2]	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;
[1.3]	retrieving content and manifest files from a content delivery server;
[1.4]	detecting content encryption key rotation boundaries between periods of use of different content encryption keys in decrypting retrieved content;
[1.5]	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
[1.6]	applying the second key for content decryption after the key rotation boundary is reached.
[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.
[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.
[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.

Element	Claim Language
[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.
[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.
[7]	The method of claim 1, wherein subsequent content encryption key identifiers are predictable based on a predetermined known progression.
[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.
[9]	The method of claim 7, further comprising: the identifier being calculated as an expected wall clock time for applying the next content encryption key based on a fixed periodic content encryption key expiration interval.
[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.
[11]	The method of claim 10, wherein the token-based technique employs white box encryption to encrypt the token.
[12]	The method of claim 1, wherein actual content encryption key rotation boundaries are detected based on real-time in-band notifications.

Element	Claim Language
[13]	The method of claim 12, wherein content encryption key rotation boundaries are detected based on notifications in an unencrypted header portion of the encrypted content files.
[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.
[15]	The method of claim 12, wherein content encryption key rotation boundaries are detected based on a notification embedded in the file name of the encrypted content file.
[16.P]	A computerized device operable as a client for handling secure distribution of content, comprising:
[16.1]	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, wherein the processors are operative to execute the computer program instructions from the memory to cause the computerized device to:
[16.2]	initiate a media playback request and receive a playback request response;
[16.3]	parse content information from the playback request response, the content information including content encryption keys, content encryption key identifiers, and content encryption key expiration times;
[16.4]	retrieve content and manifest files from a content delivery server;
[16.5]	detect content encryption key rotation boundaries between periods of use of different content encryption keys in decrypting retrieved content;

Element	Claim Language
[16.6]	issue 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
[16.7]	apply the second key for content decryption after the key rotation boundary is reached.
[17]	The computerized device of claim 16, 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.
[18]	The computerized device of claim 17, wherein the computer program instructions further cause the computerized device to time the prefetching of a next un-retrieved content encryption key based on an expected expiration of the content encryption key currently being used.
[19]	The computerized device of claim 16, wherein subsequent content encryption key identifiers are predictable based on a predetermined known progression.
[20]	The computerized device of claim 16, wherein actual content encryption key rotation boundaries are detected based on real-time in-band notifications.

Apple Inc. (“Petitioner” or “Apple”) petitions for *Inter Partes* Review (“IPR”) of claims 1-20 (“the Challenged Claims”) of U.S. Patent 9,313,178 (“’178 patent”).

I. REQUIREMENTS FOR IPR UNDER 37 C.F.R. § 42.104

A. Standing

Apple certifies that the ’178 patent is available for IPR. Apple has not been served with a complaint for infringement of the ’178 patent. Apple is not barred or estopped from requesting this review on the grounds identified in Section I.B.

B. Challenge and Relief Requested

Apple submits that this Petition demonstrates a reasonable likelihood of prevailing with respect to at least one Challenged Claim, and respectfully requests institution of IPR and cancellation of all Challenged Claims as unpatentable on the grounds set forth in the table below. Additional explanation and support for each ground is set forth in the Declaration of Dr. Aviel Rubin. (APPLE-1003), referenced throughout this Petition.

Ground	Claims	Basis for Rejection (35 U.S.C. § 103)
1A	1-4, 6-7, 12-13, 16-20	Peterka, Bocharov
1B	7-9, 14-15, 19	Peterka, Bocharov, Peterka308, Chen
1C	5	Peterka, Bocharov, Balraj
1D	10	Peterka, Bocharov, Kelly
1E	11	Peterka, Bocharov, Kelly, Eisen

The '178 patent was filed on April 30, 2014, and claims priority through an intervening application to a provisional patent application filed on June 23, 2011. For purposes of the analysis herein, and without acquiescence, Petitioner treats June 23, 2011 as the earliest effective filing date ("Critical Date") of each of the Challenged Claims in the IPR.

Each of the prior art references applied in Grounds 1A-1E qualifies as prior art to the '178 patent on at least the bases shown below:

Reference	Filed	Published	Pre-AIA 35 U.S.C. §102 Prior Art Basis
Peterka	10/26/2001	11/21/2002	§102(a)-(b), (e)
Bocharaov	03/16/2009	09/16/2010	§102(a), (e)
Peterka308	08/22/2007	10/30/2008	§102(a)-(b), (e)
Chen	---	05/12/2004	§102(b)
Balraj	04/06/2009	10/08/2009	§102(a)-(b), (e)
Kelly	12/22/2004	06/23/2005	§102(a)-(b), (e)
Eisen	05/25/2009	03/17/2011	§102(a), (e)

None of the references applied in Grounds 1A-1E were previously cited or applied in a rejection during prosecution of the '178 patent. Apple respectfully submits that discretionary institution denial under 35 U.S.C. § 325(d) would be unwarranted under these circumstances. *Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 (PTAB Feb. 13, 2020) (precedential).

II. SUMMARY OF THE '178 PATENT

A. Brief Description

The '178 patent generally describes “over-the-top (OTT) media delivery and more specifically [] encryption key rotation for live streaming media.” APPLE-1001, 1:15-17; APPLE-1003, ¶41. For example, the specification discloses a server that streams to a client device segments of media content that are each encrypted with a different content encryption key. *Id.*, 1:38-42, 1:63-65. The client communicates with a licensing server during the stream to pre-fetch content encryption keys ahead of key rotation boundaries, and the client then applies a retrieved key to decrypt a corresponding segment of the media stream. *Id.*, 11:3-25, 11:38-62. “The transitioning between use of different keys is also referred to ... as key ‘rotation’.” *Id.*, 1:63-65.

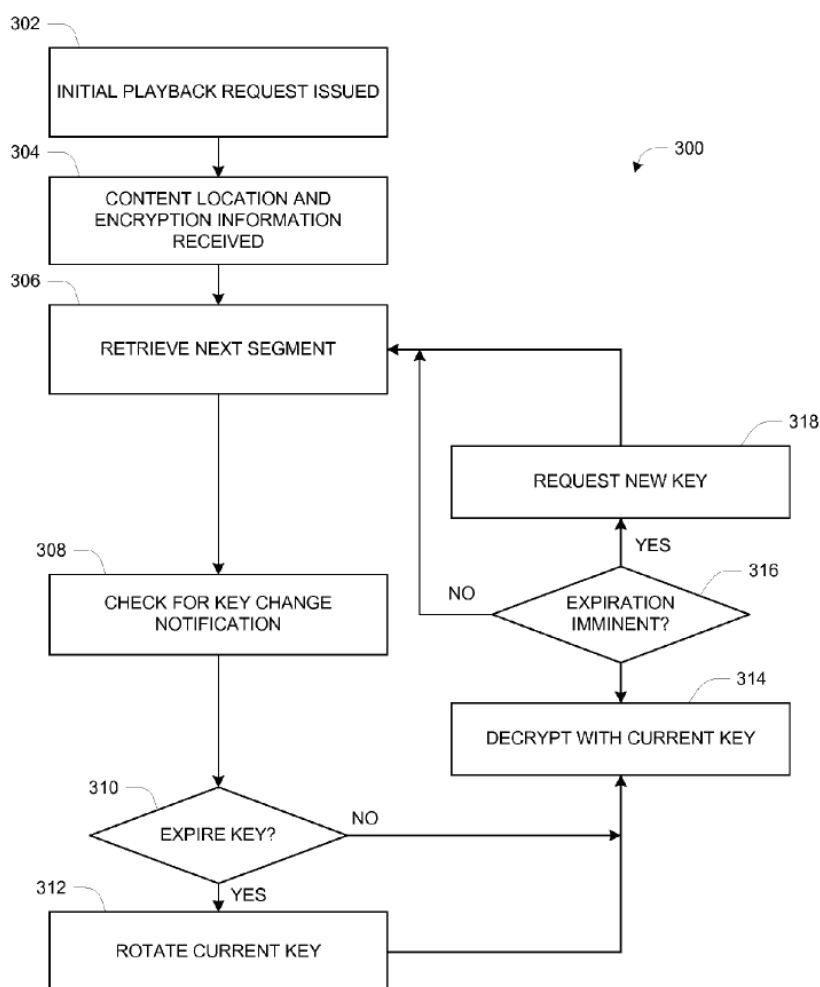


Fig. 3

APPLE-1001, FIG. 1; *generally id.*, 9:47-11:62.

B. Summary of the Prosecution History

The file history of the '178 patent reveals minimal substantive interactions between the applicant and the Examiner during original prosecution. *See* APPLE-1002; APPLE-1003, ¶42. The Examiner identified all original claims 1-20 as allowable in a first action dated May 26, 2015. *Id.*, 61-65. The Examiner cited one reference in this action, i.e., U.S. Pub. 2005/0060316 to Kamath et al., but did not

substantively comment on the reference. *Id.*, 65. The Examiner subsequently allowed the application in a notice dated October 28, 2015. *Id.*, 19-25. The Examiner's reasons for allowance stated "the prior art of record (Kamath 20050060316) does not teach detecting content key rotation boundaries and issuing requests to a license server ahead of the key rotation boundaries." *Id.*, 24.

C. Level of Ordinary Skill in the Art

For purposes of this IPR, Petitioner submits that a person of ordinary skill in the art at the time of the alleged invention ("POSITA") would have had a Bachelor's degree in computer science, computer engineering, or a related field, and 2-3 years of practical engineering experience, including experience designing or researching information security systems that employ cryptographic keys to encrypt/decrypt digital data. APPLE-1003, ¶¶23-26. Additional education could substitute for professional experience, or significant experience in the field could substitute for formal education. *Id.*

III. CLAIM CONSTRUCTION UNDER 37 C.F.R. §§ 42.104(b)(3)

All claim terms should be construed according to the *Phillips* standard. *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005); 37 C.F.R. §42.100.

Based on the prior art's description of the claimed elements being similar to that of the '178 patent specification, no formal claim constructions are presently necessary since "claim terms need only be construed to the extent necessary to

resolve the controversy.” *Wellman, Inc. v. Eastman Chem. Co.*, 642 F.3d 1355, 1361 (Fed. Cir. 2011); APPLE-1003, ¶¶43-44.

Petitioner reserves the right to respond to any constructions offered by Patent Owner or adopted by the Board in accordance with due process. Furthermore, Petitioner is not conceding that the Challenged Claims satisfy all statutory requirements including those under 35 U.S.C. § 112.

IV. THE CHALLENGED CLAIMS ARE UNPATENTABLE

A. [Ground 1A]: Peterka in view of Bocharov

1. Peterka

Peterka describes systems for securely delivering streaming media content to authorized devices over a network. APPLE-1004, [0038], [0006]-[0012], Abstract, FIG. 1; APPLE-1003, ¶¶45-56. To provide customers with different options for streaming media, Peterka’s system divides streams into “segments,” and encrypts each segment with a different content key. APPLE-1004, [0099] (“divide a single program event or an entire service into purchasable segments”), [0101] (the “content key” is “used to encrypt the content itself” and “should change at least as often as the [program segment key]”), [0091]-[0093], [0122]-[0125], FIGS. 7A-7B; APPLE-1003, ¶¶46-47. To playback an encrypted segment of the media stream, the client must obtain the corresponding “content key” for that was used to encrypt the segment, and must apply the content key to decrypt the segment before it can be

played. *Id.*, [0127] (“utilize the content key to decrypt the encrypted program content”), [0091]-[0093], FIG. 7A.

To protect content keys from unauthorized use, Peterka teaches that the content keys themselves can be encrypted by other, higher-level (secondary) keys. *Id.*, [0094]-[0104]. For authorized users, the system issues (1) an encrypted version of the content key for a given segment of media content and (2) at least one secondary key (e.g., UK, PK, PSK, or SK). *Id.* The secondary key is a key that is used to encrypt the content key (or another key) before the content key is distributed to users, and the user’s device can subsequently apply the secondary key to decrypt the encrypted content key (or to decrypt another encrypted key). *Id.* Peterka provides additional detail about the content key in paragraph [0101] and the program segment key (PSK), program key (PK), service key (SK), and unique key (UK) in paragraphs [0094]-[0104]; APPLE-1003, ¶¶48-50.

Peterka thus discloses that the PSK, PK, and SK are shared keys distributed to the client devices of potentially all users within a corresponding distribution scheme (e.g., subscription, pay-per-view, or pay-by-time). A same encrypted version of the content key can be distributed to all users within a corresponding distribution scheme, since all such users can apply their respective secondary key (e.g., PSK, PK, or SK) to decrypt the content key. *Id.*, [0094]-[0104]. On the other hand, shared secondary keys are unnecessary when the system delivers an encrypted

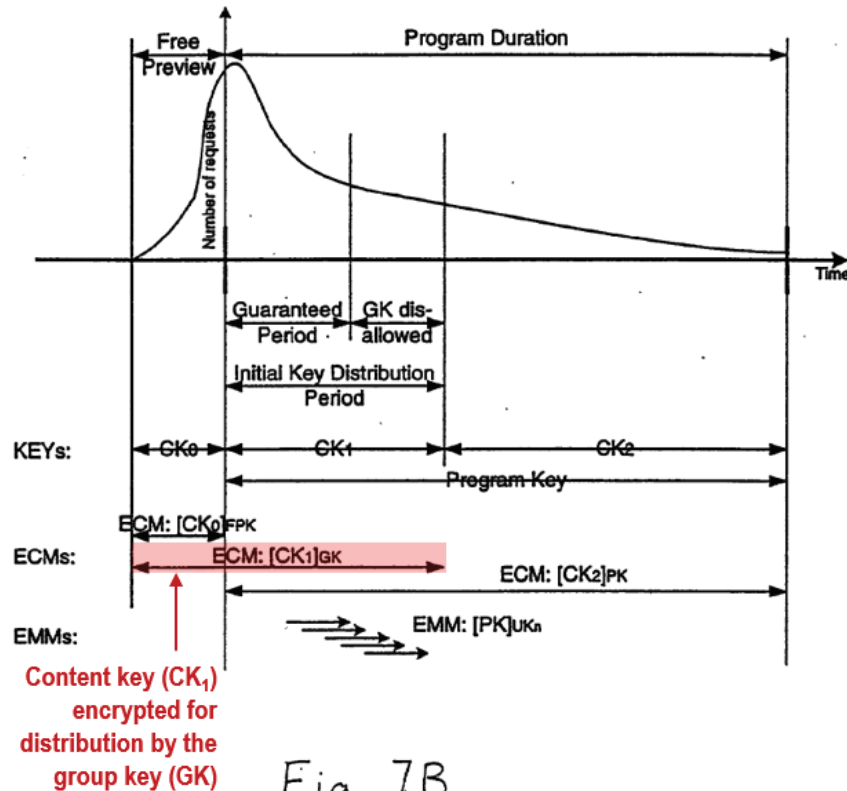
version of the content key that is specific to an individual user by encrypting the content key with that user's unique key (UK). *Id.*, [0095], [0104] ("only keys distributed to individual clients (shown as keys encrypted under the UK) using the UK are delivered in a unicast fashion"), [0108] ("a unique key for a particular client could be used to distribute the new key to that particular client"), [0125] ("Alternatively, CK2 may be encrypted with the UK and delivered in the form of an EMM directly to the client").

Peterka further describes techniques for implementing "free" and "initial" viewing periods during which users may access an initial portion of encrypted media content before a user decides to purchase the content (or otherwise before keys encrypted by a user's unique key have been distributed). *See id.*, [0005], [0059], [0083]-[0093], [0122]-[0134], FIGS. 3-6, 7A-7B. The free preview period is implemented by distributing a "free preview key" (FPK) to users in advance of a program's start time. *Id.*, [0059], [0103]. The FPK can be applied either to decrypt a content key encrypted by the FPK, or the FPK can be applied to decrypt the media content directly during the free preview period. *Id.*, [0086], [0123]-[0124], [0129].

The "initial" viewing period then allows users who have requested to continue consuming the content beyond the free preview period. *Id.*, [0126]-[0133]. During the initial viewing period, users can apply a "group key" (GK) either to decrypt a content key encrypted by the GK for the initial viewing period or to directly decrypt

the media content during the initial viewing period. *Id.*, [0123]-[0124], [0128]-[0129].

Figure 7B, for example, shows how a group key may be applied during the initial key distribution period to decrypt content key CK1:



APPLE-1004, FIG. 7B (annotated), [0122]-[0125].

Peterka's system streams encrypted media content to client devices in packets, e.g., RTP packets. *Id.*, [0117]-[0120]. Servers further distribute keys to client devices in Entitlement Control Messages (ECMs) or Entitlement Management Messages (EMMs). *Id.*, [0104], [0121]-[0125], FIG. 7B; APPLE-1003, ¶¶52-53.

In some embodiments, Peterka's system implements a "pull model" that

leaves client devices with the responsibility for requesting keys from a server at appropriate times ahead of a segment boundary. *See id.*, [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.”), [0106], FIG. 8; APPLE-1003, ¶¶54-56.

2. Bocharov

Bocharov discloses “[a] smooth streaming system [that] provides a stateless protocol between a client and server” for distribution of streaming media content. APPLE-1005, Abstract, [0001], [0007], [0053], FIGS.3-5; APPLE-1003, ¶¶57-60. Figure 5 is a flow diagram that depicts interactions between an origin server 510, an encoder 505, and a client 515 in Bocharov’s system:

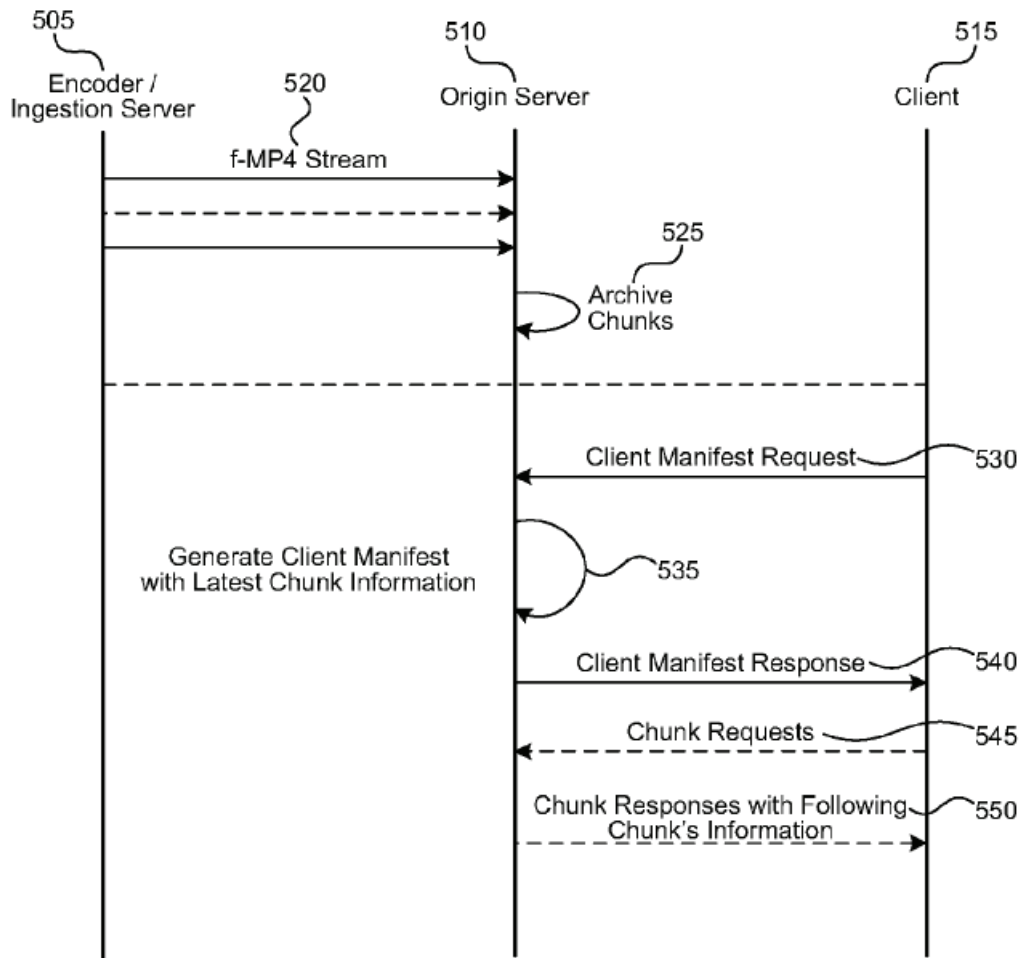


FIG. 5

APPLE-1005, FIG. 5.

As shown in Figure 5, “[t]he encoder 505 continuously provides media data 520 to the origin server 510.” *Id.*, [0053]. “The media data may include fragments of an MP4 stream based on a live event, for example.” *Id.* “The origin server 510 archives 525 each media fragment, such as to a local data store.” *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 id.*, [0006]-[0007], [0014]-[0016], [0021]-[0031], [0049]-[0057], [0060].

Bocharov teaches that the “client manifest” includes information useful for the client to request media fragments from a server for either pre-recorded or live programming. For example, the manifest can provide timestamps for each fragment archived up to the point of the request, and the client uses the timestamp to compose a URL for requesting the fragment from the server. *Id.*, [0030]-[0031]. The manifest can further identify different available codings of the media content. Thus, if a program has been encoded in multiple bit rates, the client can reference the manifest to “begin requesting fragments in whichever encoding the client chooses.” *Id.*, [0049]. “For example, the client may initially select a low bit rate encoding and select a 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 id.*, [0023]-[0025], [0031]-[0031], [0049], [0052], [0057], [0060].

3. Obviousness Based on Peterka-Bocharov Combination

(a) Overview of Combination

As described above, Peterka describes techniques for distributing streaming

media content to client devices over a network. *Supra*, Section IV.A.1; APPLE-1003, ¶¶68-75. According to Peterka, servers are configured to stream encrypted media content to client devices, and the client devices in turn apply decryption keys to recover clear versions of the content for playback. APPLE-1004, [0002], [0004], [0038]. Peterka explicitly notes that the transmission network “can be implemented in a variety of ways,” and can employ various protocols suitable for directly streaming media content or facilitating streaming (e.g., UDP, RTP, RTCP, IGMP, TSP, SAP/SDP). *Id.*, [0043]; *see also id.*, [0038]-[0043], [0117] (“refers to use of an RTP packet for distributing program content; however, it could equally apply to other protocols utilized in distributing content”), FIGS. 1, 14.

Peterka also teaches that streaming content can be offered at different “qualit[ies]” (i.e., “bit rates”). *Id.*, [0054]. For example, “[t]he client may report the highest bit rate it can consume and be offered to purchase the content at that quality or less.” *Id.*, [0054]. However, beyond the brief description of these features in paragraph [0054], Peterka offers little additional detail as to how multi-bit-rate streaming could be implemented or how a client could “adjust the bit rate perceived by the user,” for example. *Id.*; APPLE-1003, ¶69.

Bocharov demonstrates that techniques for managing multi bit-rate streaming services were not new in the ’178 patent. APPLE-1003, ¶70. As discussed above, Bocharov discloses techniques that allow a client to obtain fragments of media

content “in whichever encoding the client chooses.” APPLE-1005, [0049]; *see also id.*, [0007], [0014], [0016], [0049]-[0062], [0060], FIG. 5; *supra*, Section IV.A.2. Bocharov’s solution involves issuing manifest files to clients, where the manifest specifies different encodings available for streaming media fragments. *Id.*

According to Dr. Rubin, it would have been obvious to a POSITA by the Critical Date of the ’178 patent to combine the teachings of Peterka and Bocharov to achieve predictable benefits. APPLE-1003, ¶¶71-75. For example, as shown in Figure 5 of Bocharov, it would have been predictable for a client device in a Peterka-Bocharov combination to issue a client manifest request to a server, and to obtain a manifest in response. APPLE-1005, [0053], FIG. 5. Consistent with Bocharov’s teachings, the manifest can include information that allows a client to request delivery of streaming media content, such as information indicative of a URL for the media content. *Id.*, [0030]-[0031]. As described in Bocharov, the manifest can also identify different encodings such that the client could request delivery of the streaming media according to a particular encoding (e.g., corresponding to a desired “quality” or “bit rate”). *Id.*, [0030]-[0031], [0048]-[0050]. Further, in some implementations (although not required), the manifest provides a list of fragments in the media stream with information indicative of a URL for each fragment. *Id.* Individual client devices can then reference the manifest to request timely delivery of fragments to maintain a continuous media stream. *Id.*

Multiple reasons would have led a POSITA to combine the teachings of Peterka in view of Bocharov in a manner like that described in the preceding paragraph. APPLE-1003, ¶72. To start, the teachings of Peterka and Bocharov are both directed to highly similar systems for streaming media content. *See supra*, Sections, IV.A.1-2; APPLE-1004, Abstract, [0002], [0004], [0038]-[0043], [0117], FIGS. 1, 4; APPLE-1005, [0001], [0007], [0014]-[0016]; APPLE-1003, ¶72. Both references describe technologies that are particularly useful for live streaming, and each provides solutions for managing loads that can result from large numbers of users requesting access to live streamed content at nearly the same time. *E.g.*, APPLE-1004, [0064] (“there will be a large number of clients requesting access to the same event at the same time” and this “will generate a large load on the system within a relatively small time window”), [0126] (“If the population is very large, a server will be unable to distribute keys to all participants instantly”), [0083]-[0093] (“free preview period”), FIGS. 7A-7B; APPLE-1005, [0014] (“Caching reduces the load on the server and allows more clients to view the same content at the same time.”), [0052]. Bocharov’s use of manifests to regulate client content delivery would thus complement Peterka’s techniques for distributing keys to many users to access encrypted media content, thereby improving the resulting system’s ability to handle large loads with respect to both content delivery and key distribution. APPLE-1003, ¶72. Both references also stream similar types of media (e.g., video,

MP4 files) and apply segmentation to divide the media into smaller chunks (e.g., fragments) for distribution. APPLE-1004, [0080], [0119]; APPLE-1005, [0016], [0020], [0034], [0038]-[0039]; APPLE-1003, ¶72.

Moreover, applying Bocharov's suggested use of manifests in combination with Peterka would have provided an effective solution for distributing multiple versions of a media program having different encodings/qualities/bit rates. APPLE-1003, ¶73. Peterka's original disclosure already acknowledged the benefit of providing streams of different qualities that could accommodate changing user demands and network conditions, and Bocharov's manifest would have been predictable to provide clients with the information needed to more readily select and switch between media streams having different encodings/qualities/bit rates. APPLE-1004, [0054]; APPLE-1005, [0049]; *see also id.*, [0007], [0014], [0016], [0049]-[0062], [0060], FIG. 5. Bocharov even describes the use of manifests to improve tiered quality purchasing similar to Peterka's "pay-by-quality" model—e.g., explaining that "the system may enable or disable switching to particular bit rates based on whether the conditions of the payment model have been met" and "[t]his information may be included in the manifest provided to the client." APPLE-1005, [0060]; APPLE-1003, ¶73.

Furthermore, it would have been predictable for some implementations of a combination based on Peterka and Bocharov to issue client manifests to enable a

“stateless protocol” for streaming media distribution. *See* APPLE-1005, [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.”). A POSITA would have been motivated to adopt a “stateless protocol” like that described in Bocharov to achieve the benefits of such a protocol expressly articulated in Bocharov. APPLE-1003, ¶74. For example, Peterka’s original solution relied on protocols such as RTP to carry streaming media packets to clients and RTCP to provide stateful control between servers and clients. APPLE-1004, [0043], [0117]-[0121]; *see also* APPLE-1005, [0001]-[0004]. Bocharov recognized, however, that “[t]he stateful connection between client and server creates additional server overhead, because the server tracks the current state of each client,” and “[t]his also limits the scalability of the server” and can prevent clients from reacting “quickly ... to changing conditions.” APPLE-1005, [0004]. Applying Bocharov’s suggested use of manifests and stateless control in combination with Peterka would at least partially mitigate these problems. APPLE-1003, ¶74. For example, stateless control can eliminate the need for a separate control channel (e.g., RTCP channel) and “provides a more scalable streaming media server without tracking client state and with an increased likelihood that clients will receive media with lower latency from a cache server local to the client.” APPLE-1005, [0007], [0014] (“the client can, for

example, switch quickly to fragments of a different bit rate or playback speed based on network conditions”), [0052] (“This allows the server to focus resources on other tasks and dramatically increases scalability. For well-attended live events, this means that many more viewers can watch the event.”); APPLE-1003, ¶75.

(b) Claim Element Analysis

Peterka and Bocharov would have rendered obvious claims 1-4, 6-7, 12-13, and 16-20, and the combination (*supra*, Section IV.A.3(a)) discloses each limitation of these claims as described below.

Element [1.P]

To the extent the preamble is a limitation, the Peterka-Bocharov combination provides a method for handling secure distribution of content. *Supra*, Section IV.A.1; APPLE-1004, [0012] (“a method of providing program material to multiple clients”), [0094] (“Under various distribution methods such as pay by time, pay per view, subscription based, etc., encryption keys can be distributed to clients to facilitate reception of the program content.”), [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.”), FIGS. 1, 3; APPLE-1003, ¶¶76-77.

Element [1.1]

Peterka and Bocharov disclose a client device initiating a media playback request and receiving a playback request response. APPLE-1003, ¶¶78-81.

For example, Peterka teaches that clients may request to purchase live or otherwise scheduled media content, and such requests can be made at various times relative to the scheduled content transmissions:

Purchase Timing—Clients may purchase content at different times: Ahead of time—The client decides to purchase the content significantly ahead of time. Such a purchase may be associated with the entire service, such as Subscription, rather than a single program.

Just before, or during, the program—The client decides to purchase the content a short period of time before the beginning of the content, or during the content very near to the start.

APPLE-1004, [0055]-[0056]; *see also id.*, [0091] (“FIG. 7A illustrates a plot showing the number of **requests** for a program versus the program duration”), [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.”), [0110] (“each client **requests** a new key and is individually responded to by the key distribution server”), [0131] (“all clients who **request** access to an event (e.g., they request a program key for a pay per view event) during the

‘guaranteed period’ will be guaranteed to receive the content without interruption”), [0137] (“the origin content server analyzes the client’s *request* for program content”), [0143] (“*request* sent by the client to the caching server”), FIGS. 7A-7B (emphases added), FIGS. 18-20.

Peterka’s servers stream content to client devices in encrypted form, and client devices under the “pull key distribution model” are configured to request distribution of decryption keys. *Supra*, Section IV.A.1; [0104]-[0106] (“pull key distribution model allows each client to actively request keys from the server”), FIG. 8. These client-initiated key requests allow the clients to obtain decryption keys and other information needed to decrypt streaming media content for playback at the client devices, and thus provide a “*media playback request*” as recited in claim 1. APPLE-1004, [0038] (“The object can be played during download if it is a streaming media or can be stored for later display.”), [0101], [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.”), [0106] (“the program content is distributed for decryption by the client utilizing the cryptographic key that was requested”); APPLE-1003, ¶¶79-80. The ’178 patent specification likewise connects “media playback requests” to key requests, and does not require a media playback request to inspire retrieval of the encrypted content itself. *Cf.* APPLE-1001, 2:15-30, 9:47-56, 10:25-38, FIG. 3 (distinguishing steps

302 and 306); *see also id.*, Claim 1 (Element [1.3] content retrieval recited separately from media playback request and response).

After initiating a media playback request (e.g., requesting a next set of keys for decrypting a media program), Peterka explains that the client receives the requested keys and related information in one or more Entitlement Management Messages (EMM) or Entitlement Control Messages (ECMs) (***playback request response***). APPLE-1004, [0104], [0121] (“the type of key included in the message, such as a content key, a group key, a program segment key, a service key, etc.”), [0122]-[0125], FIG. 7B; APPLE-1003, ¶81. For example, under the pull-key distribution model, clients can request any of the keys represented in Table 1 (Peterka, [0104]) (e.g., PK, PSK, CK, GK, FPK), and the keys “are distributed to the client in response to the request by the client.” APPLE-1004, [0106], FIG. 8; APPLE-1003, ¶81. To access the free preview and initial viewing periods of a program, a client can request and obtain a free preview key (FPK) and one or more group keys (GKs) in advance of a program start time. *Id.* Keys for additional segments or a duration of the program can be requested in advance of the segments during the course of the program. *Id.*; *see also supra*, Section IV.A.2 & APPLE-1005, Abstract, [0014]-[0015] (describing “client requests fragments of media from a server that include a portion of the media”), [0050]-[0051] (client requesting media fragments for playback and receiving fragments for playback in response).

Element [1.2]

As discussed above, Peterka discloses a server that returns to a client device one or more Entitlement Control Messages (ECMs) or Entitlement Management Messages (EMMs) (***playback request response***) in response to a key request or other request to access/play at least a portion of a media stream (***media playback request***). *Supra*, Element [1.1], Section IV.A.1; APPLE-1003, ¶¶82-91. ECMs and EMMs contain information responsive to the request, and a POSITA would have understood that the client would parse an ECM/EMM to extract and distinguish among the different pieces of information contained in the message. APPLE-1004, [0104], [0121]-[0125], FIG. 7B; APPLE-1003, ¶82. Specifically, Peterka describes the content of ECMs/EMMs as follows:

[E]ntitlement management messages and entitlement control messages can be sent from a server to client computer. One embodiment of the invention provides a format for such messages. Under this format, a sequence number or a time stamp to protect against replay attacks is provided. Furthermore, in another field of the EMM and ECM messages, a keyed message authentication code (MAC), or a public key digital signature for authentication is provided. Note that neither the keyed-hashing for message authentication (HMAC) nor the signature can be verified until a client performs the key request exchange. Yet another field would include the type of key included

in the message, such as a content key, a group key, a program segment key, a service key, etc. Furthermore, a field would be provided for the type of key used to encrypt the key in the message. Thus, a unique key could be indicated for a message transmitting a service key, for example. 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. Also, a user identification, which is often needed when multiple EMMs are delivered in a single multicast message, can be provided.

APPLE-1004, [0121].

Peterka's disclosed ECM/EMM messages provide each type of "*content information*," as recited in Element [1.2]:

- **Content Encryption Key** – ECM/EMM messages carry keys to client devices, including a "content key" (CK) (*content encryption key*). APPLE-1004, [0121] ("the type of key included in the message, such as a content key"), [0101] ("Content Key (CK): This key is used to encrypt the content itself."). For the free preview and initial viewing periods, the free preview key (FPK) and group key (GK) distributed through ECM/EMM messages are also "*content encryption keys*." APPLE-1004, [0086] ("the FPK may be used to directly encrypt the initial portion of the content, in which case the initial content key is the same in value as the FPK"),

[0123], [0124] (“The value of GK already possessed by the plurality of clients may be taken as CK1.”), FIG. 7B; APPLE-1003, ¶¶83-84.

- **Content Encryption Key Identifier** – ECM/EMM messages provide several pieces of information that correspond to a “*content encryption key identifier*” (CEK ID) as recited in claim 1. APPLE-1003, ¶85. **First**, the field that identifies the “type of key included in the message” provides a CEK ID. APPLE-1004, [0121]; APPLE-1003, ¶85. When the ECM/EMM carries a CK, FPK, or GK, the “type” field would identify the key as a content encryption key since each of these key types is used for content encryption. *Id.* Furthermore, Peterka teaches that some programs include just a single free preview period (which is encrypted/decrypted by a single FPK), a single initial viewing period (which is encrypted/decrypted by a single GK), and a single program segment following the initial viewing period. *See, e.g.*, APPLE-1004, [0122]-[0133], FIGS. 7A, 7B. When a program includes just a single one of any of these periods, the “type” identifier for the corresponding key (FPK, GK, or CK) also uniquely identifies the key from any other content encryption key applied in the program. APPLE-1003, ¶85. **Second**, the “parity” value provides a CEK ID that the client references to distinguish among multiple content encryption keys in its possession and to select a particular key to apply against the content for decryption at a given time (e.g., by matching the “parity” value from an ECM/EMM message with a “parity” value in a content stream packet). APPLE-1004, [0117]

(“A signaling method can be used to indicate the implementation of a new key. For example, a predetermined bit can be used 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. Thus, a client can check the predetermined bit in a packet and determine the proper content key to use. As one example, if a single bit is used, a ‘1’ could be used to indicate the current content key already in use, while a ‘0’ could be used to indicate that the new content key should be used.”), [0118] (“header could include at least the content key parity bit to indicate key changes”), [0119]-[0120], [0121] (ECM/EMM message includes “a key parity bit matching the parity bit in the RTP packet”); APPLE-1003, ¶85. Peterka’s example uses a single parity to distinguish successively applied content encryption keys, but the disclosure “if a single bit is used” suggests that longer, multi-bit parity values could alternatively be applied to uniquely distinguish among even more keys. APPLE-1004, [0117], FIG. 14; APPLE-1003, ¶85. **Third**, Peterka discloses that each ECM/EMM message includes “a sequence number or a time stamp to protect against replay attacks.” APPLE-1004, [0121]. Since the sequence number or time stamp is unique to each ECM/EMM, and each ECM/EMM carries a different content encryption key (e.g., CK, FPK, or GK), the sequence number or time stamp would provide a unique CEK ID. APPLE-1003, ¶85. **Fourth**, Peterka discloses that each ECM/EMM message includes “a keyed message authentication code (MAC), or a public key digital

signature for authentication.” APPLE-1004, [0121]; APPLE-1003, ¶85. Since the keyed MAC or digital signature is unique for each ECM/EMM, and each ECM/EMM carries a different content encryption key (e.g., CK, FPK, or GK), the keyed MAC or digital signature would provide a unique CEK ID. APPLE-1003, ¶85.

- **Content Encryption Key Expiration Times** – Peterka discloses that an ECM/EMM message includes a field indicating “the time remaining in the lifetime of the key.” APPLE-1004, [0121]. The time remaining in the lifetime of the key corresponds to a “*content encryption key expiration time*,” particularly when the ECM/EMM message carries a content encryption key (e.g., CK, FPK, or GK). APPLE-1004, [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.”), [0106] (“In block 818, the key and its expiration time are distributed to the client in response to the request by the client.”); *see also id.*, [0114] (“monitor the expiration time” of the PSK for which the CK “changes at least as often” (*see id.*, [0101])), FIG. 8; APPLE-1003, ¶86.

Peterka also discloses or renders obvious a “*playback request response*” having “*content information*” that provides *multiple* content encryption keys, content encryption key identifiers, and content encryption key expiration times. APPLE-1003, ¶87. For example, Peterka subdivides streaming media into

“segments” and applies a different content encryption key to each segment. APPLE-1004, [0099] (“divide a single program event or an entire service into purchasable segments”), [0101] (the “content key” is “used to encrypt the content itself” and “should change at least as often as the [program segment key]”), [0091]-[0093], [0122]-[0125], FIGS. 7A-7B. In this context, Peterka describes “a key management approach where each rabbit is given keys to the current as well as the next segment.” *Id.*, [0080]. Figure 7B also shows the distribution of two content keys CK₀ and CK₁ in separate ECMs during the free preview period, and two content keys CK₁ and CK₂ in separate ECMs during the initial key distribution period. *Id.*, FIG. 7B. Thus, Peterka expressly teaches techniques for providing keys for multiple streaming segments to a client device ahead of a segment boundary. According to these teachings, a client could request access to an ongoing media stream in a media playback request (*see supra*, Element [1.1]), and be provided with a playback request response comprising the keys to the current as well as the next segment. *Id.*; APPLE-1003, ¶87. The response with the keys for both segments would be provided in ECM/EMM messages containing key identifiers and expiration times as described above. APPLE-1004, [0121]; APPLE-1003, ¶87.

Peterka indicates that each key distributed to a client is sent in its own respective ECM or EMM message, and thus a POSITA would have understood that the content encryption key for the current segment and the next segment would be

distributed with separate key identifiers in separate ECM or EMM messages. APPLE-1003, ¶88. Indeed, paragraph [0121] repeatedly refers to the “*the* key” included in the ECM or EMM based on the understanding that each ECM or EMM contains just a single key. APPLE-1004, [0121]; *see also id.*, [0104] (describing how multicast messages are still distributed in “multiple EMMs encrypted with different UKs”). At minimum, it would have been obvious to distribute each content encryption key in a separate ECM or EMM message that contained a respective set of metadata (e.g., key identifiers and expiration times) in order to reduce the complexity of parsing ECM/EMM messages at a client device. For example, such distribution in separate messages would obviate any need for the client to perform additional processing to detect whether an ECM/EMM contained multiple keys or resolve how to apply or differentiate the metadata for each key as may otherwise occur if multiple keys were combined in a single ECM/EMM message. APPLE-1003, ¶88. To this point, paragraph [0121] of Peterka notes that a “field can be provided for the time remaining in the lifetime of the key.” APPLE-1004, [0121]. Since the keys for two successive segments in a media stream would naturally have different lifetime values, it would have been predictable and convenient to distribute the respective content encryption keys for each segment in different ECMs or EMMs to more clearly and simply associate each key with the respective lifetime values (and other metadata) that apply to the specific key. APPLE-1003, ¶88.

To the extent Peterka's description in paragraph [0080] does not expressly refer to *content encryption keys* (e.g., CKs, FPK, GK) for the current and next segment, a POSITA would have understood Peterka's teaching here to encompass content encryption keys since they are among the "given keys" that the client obtains for each segment (and are indeed necessary for the client to decrypt content in each segment). APPLE-1004, [0101], [0091]-[0093], FIG. 7A; APPLE-1003, ¶89. At the very least, it would have been obvious to distribute content encryption keys for both the current and next segments of a media stream in a playback request response in at least some distribution models suggested by Peterka. APPLE-1003, ¶89. For example, under the "pay-per-view" (PPV) model, Peterka provides an option for the "content key" (CK) and "group key" (GK) both to be distributed to a client encrypted by the client's own unique key (UK).¹ APPLE-1004, [0101]-[0102], [0104] (Table 1). In this model, the client has no need for other keys such as a PSK or PK to decrypt the content encryption key. *Id.* An ordinary application of Peterka's teaching in paragraph [0080] to the user-specific content encryption key PPV distribution model would have predictably entailed distributing content encryption keys to the client for both the current and next segments in response to a client's

¹ The CK and GK are both "content encryption keys" used for encryption and decryption of streaming media. APPLE-1004, [0101], [0124].

media playback request (such as a GK for the current segment and a CK for the next segment, or respective CKs for the current and next segments). APPLE-1003, ¶89. Indeed, Peterka expressly acknowledges that GKs and CKs are among the types of keys that can be requested by a client device and returned in ECM/EMM message responses. APPLE-1004, [0104]-[0105] (“models for distributing the keys outlined above” include a “pull model”), [0106], [0121], FIG. 8. A POSITA would have sought to return content encryption keys for successive segments in response to a client’s request to reduce the “load” on the caching server at segment boundaries. *See* APPLE-1004, [0080]; APPLE-1003, ¶89. By providing a client with a CK for the next segment in response to a client’s request for the current segment, the system would obviate the need for the client to issue a separate request for a CK for the next segment and would reduce the volume of requests that the server needs to respond to ahead of the segment boundary. *Id.* Additionally, it would have been beneficial for the client device to have the next CK in case the server was unable to distribute the CK for the next segment in sufficient time for the client to apply the next CK at the start of the next segment (which could lead to disruption in client’s ability to decrypt and playback the streaming media). *Id.* Peterka teaches that the multiple keys would be distributed in messages such as EMMs/ECMs that each contain the respective components recited in claim 1 (as described above). APPLE-1004, [0104], [0121]; APPLE-1003, ¶89.

For similar reasons, a POSITA would have found it obvious based on Peterka's teachings to implement a "pay-by-time" (PBT) event in which a server returned user-specific content encryption keys to the client for both the current and next segments in response to a client's media playback request. APPLE-1003, ¶90. Paragraph [0080] is already described in the context of a quasi-PBT model, and user-specific CK encryption (in which the CK is encrypted by an individual client's UK) would have been obvious to try in a PBT event given the finite set of options for encrypting the CK in Peterka (e.g., Table 1 shows the CK can be encrypted only by the PK, FPK, GK, UK, PSK, or SK). APPLE-1004, [0104]; *see also id.*, [0101] (describing non-limiting *examples* for CK distribution: "[i]t can be distributed in several ways"). User-specific CK encryption for PBT events also would have been beneficial to reduce the total number of keys needed to distribute to clients, as separate PSK distribution would be unnecessary if the CK for each segment was directly encrypted by individual clients' respective UKs. APPLE-1003, ¶90.

As another example, a client device can obtain multiple content encryption keys (and associated identifiers and expiration times) in a playback request response by issuing a request that results in the return of both a free preview key (FPK) and a group (GK) used for content encryption. APPLE-1004, [0086], [0102]-[0104], [0124]; APPLE-1003, ¶91. The FPK and GK can be distributed to clients either through "provisioning" or in EMMs encrypted by an individual client's UK. *See*

APPLE-1004, [0102]-[0104] (Table 1), [0083]-[0085], [0127]-[0129], FIGS. 3-4, 15-16. A client would “register” with the system before keys would be “provisioned” to the client, and the registration would provide another “*media playback request*” and the messages carrying the keys another “*playback request response*.” APPLE-1004, [0136] (“a client registers with the authorization center 1920 to obtain a ticket which defines what type of content the client is entitled to obtain”), [0135]-[0144], FIGS. 19-21; APPLE-1003, ¶91. If not distributed through “provisioning,” the FPK and GK would both be returned in response to a client’s request for a particular media stream. This would occur when the FPK and GK are encrypted by the device’s UK. APPLE-1004, [0095], [0104], [0127] (“A group key may be given to clients when they request a particular caching server ...”). If a client requested the GK before the free preview period and the client had not yet received the the FPK, it would have been obvious to return both the FPK and GK to the client responsive to the request since the client would need the FPK to access and decrypt content during the free preview period that precedes use of the GK.

Element [1.3]

Peterka discloses a client device (e.g., client computer 112) that retrieves content from a content delivery server (e.g., origin 108, a content exchange 116, or caching server). APPLE-1004, [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], [0060] (“Origin Content Server (OCS)--Server at a content provider’s computer that provides content”), [0071] (OCS “becomes part of the distribution network”), [0083], [0135], FIG. 1; *supra*, Section IV.A.1. Bocharov likewise discloses a client device that retrieves content from a content delivery server, e.g., by requesting “fragments” of streaming media in a timely manner to permit continuous playback at the client device. APPLE-1005, [0007] (“the server provides uniform media fragment responses to media fragment requests”), [0015] (“client requests fragments of media from a server that include a portion of the media”), [0022] (“fragments to provide to clients based on received client requests”); *supra*, Section IV.A.2; APPLE-1003, ¶92.

Furthermore, Bocharov discloses a client device retrieving manifest files from a content delivery server. APPLE-1005, [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.”), FIG. 5; *see also id.*, [0023]-[0025], [0030]-[0035], [0049]-[0052]; *supra*, Section IV.A.2. Bocharov also teaches retrieving multiple manifest files. For example, retrieved media fragments

include updated manifests with information that may not have been available in the initial manifest. APPLE-1005, [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]. The client would also retrieve an additional manifest file when initiating retrieval of a new media stream. APPLE-1003, ¶93.

For the reasons described in detail in the overview (Section IV.A.3(a)), it would have been obvious to combine the teachings of Peterka and Bocharov such that the client device would retrieve content and manifest files from a content delivery server. *Supra*, Section IV.A.3(a); APPLE-1003, ¶94.

Element [1.4]

Peterka discloses “signaling method[s]” for signaling to the client information for the client to detect content encryption key rotation boundaries, e.g., boundaries that require use of a new content encryption key to decrypt a retrieved content from a segment of a media stream. APPLE-1004, [0116] (“[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.”), [0117] (“a client can check the predetermined bit in a packet and determine the proper content key to use”); *generally id.*, [0117]-[0121]; APPLE-1003, ¶95.

Peterka also discloses client devices configured to detect upcoming content

encryption key rotation boundaries between periods of use of different content encryption keys, based on known key expiry times that allow the client to request new keys ahead of the expiration or key rotation boundary. APPLE-1004, [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.”), [0106], [0114] (“monitor the expiration time of the program segment key and send an individual key update request before the key expires”), FIG. 8; APPLE-1003, ¶¶96-97.

Element [1.5]

Peterka discloses a client device issuing at least one request to a “**license server**” (e.g., an “origin,” “content exchange,” “caching,” or “authorization” server) ahead of a key expiration time (***key rotation boundary***) to retrieve a second “**content encryption key**” (e.g., a CK or GK) to be used after a content encryption key rotation boundary is reached. APPLE-1004, [0105]-[0106] & FIG. 8 (“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.”); *see also id.*, [0114] (example of client “monitor[ing] the expiration time” for a PSK (which the client would similarly do for a content encryption key that changes for each segment as well)); APPLE-1003,

¶98; *supra*, Elements [1.1]-[1.2], Sections IV.A.1-3(a).

The specification of the '178 patent provides no explanation as to how *multiple* requests might be issued to a license server that all pertain to retrieving the same “second content encryption key.” *Cf.* APPLE-1001, 11:21-25 (“client 110 must issue a new content encryption key request to the license server 106 to retrieve the new content encryption key”); *generally id.*, 11:3-62, FIG. 3; APPLE-1003, ¶99. Nonetheless, Peterka would have provided for issuance of multiple requests to obtain a second content encryption key, e.g., by first sending a request to initiate a “ticket request message exchange” session with a caching server and then requesting the CK itself (which may be encrypted by the UK established for the session), or by first requesting a PSK needed to decrypt a CK and then requesting the CK itself. APPLE-1004, [0095], [0099]-[0101], [0104]; APPLE-1003, ¶99. Clients also commonly re-issued requests to server if a first request failed due to a timeout, a server error, or transmission failure. APPLE-1003, ¶¶99-100 (citing corroboration APPLE-1014, -1015). To increase the likelihood that the client would have a next key available in the event of a failed request, it would have been obvious to re-issue a request for a second content encryption key to a license server ahead of a key rotation boundary if the first request failed for any of the foregoing reasons. *Id.*

Element [1.6]

Peterka discloses applying the second key for content decryption after the key

rotation boundary is reached. APPLE-1004, [0116] (“when 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] (“receive keys for decrypting the encrypted portion of the program”), [0092] (“keys are required to decrypt the remainder of the program content”), [0108] (“the program content is distributed for decryption by the client utilizing the transmitted key”); *supra*, Section IV.A.1; APPLE-1003, ¶¶101-102.

Claim [2]

As described above with respect to Element [1.2], Peterka-Bocharov discloses 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. *Supra*, Element [1.2]; APPLE-1004, [0080], [0101]-[0105]; APPLE-1003, ¶¶103-104.

Claim [3]

Peterka discloses that media streams can be divided into “purchasable segments” to allow users to pay for content by the amount of time (segments) of the stream that the user consumes. APPLE-1004, [0099], [0078]-[0082], [0101]. Each segment is encrypted by a different content key, and Peterka indicates that the segments have a defined period that yields “periodic key rotation.” APPLE-1004,

[0082] (“The client will monitor the expiration time for each pay *period* and request a new set of keys from the caching server.”), [0110] (“*periodically* given new keys using a multicast UDP message which has a new program segment key encrypted for each participant”), [0110] (“the server can issue new keys at regular intervals”). A POSITA would have understood that key rotation would occur on a periodic basis that corresponds to the periodic distribution of new keys. APPLE-1003, ¶¶105-106. At a minimum, a POSITA would have understood periodic key rotation to be a predictable option for dividing media streams into segments—particularly since Peterka already distributes keys on a periodic basis and periodic key rotation would have allowed the encryption system and decryption systems to readily determine future key rotation boundaries. *Id.* Furthermore, since users can pay for segments by “time,” periodic key rotation would have provided a fair method for users to pay costs associated with fixed-length segments defined by the periodic key rotation interval. *Id.*; APPLE-1004, [0078]-[0082], [0099]-[0101]. Dr. Rubin also observed that periodic key rotation was well known before the Critical Date of the ’178 patent. APPLE-1003, ¶107.

Peterka further discloses that key distribution messages (e.g., ECMs/EMMs) identify the expiration time of the key provided in the message. APPLE-1004, [0082], [0105] (“each client keeps track of the keys and their expiration times and actively requests new keys before the current keys expire”), [0106], [0121] (“time

remaining in the lifetime of the key”), FIG. 8; *supra*, Element [1.2]. In the context of Peterka’s disclosed “periodic” key rotation, the key expiry time (or “time remaining in the lifetime of the key”) would indicate an “expected minimum interval between periodic key rotation” since the keys are expected to expire and rotate on a fixed, periodic basis that defines a minimum (and maximum) interval between periodic key rotation. APPLE-1003, ¶108. To the extent Peterka does not expressly disclosed how the expiration times are “expressed,” it would have been obvious to express the key expiration time as recited in claim 3 to inform the client device of the period of key rotation so that the client could readily project key rotation boundaries and determine an appropriate time to request a new key ahead of the next boundary. APPLE-1004, [0105]-[0106]; APPLE-1003, ¶108. Moreover, expressing the content key expiration time in this manner would beneficially conserve bandwidth by allowing expiration times to be omitted from subsequent key distribution messages (e.g., since clients could be relied upon to calculate subsequent key expiration times based on the previously expressed expected minimum interval between periodic key rotation). *Id.*

Claim [4]

Peterka discloses a client device timing the prefetching of a next un-retrieved content encryption key based on an expected expiration of the content encryption key currently being used. APPLE-1004, [0105] (“each client keeps track of the keys

and their expiration times and actively requests new keys before the current keys expire”), [0082] (“client will monitor the expiration time for each pay period and request a new set of keys from the caching server”), FIG. 8; *supra*, Element [1.4]-[1.5], [3]; APPLE-1003, ¶¶109-110.

Claim [6]

Peterka discloses that “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.” APPLE-1004, [0105], [0106] (“In block 818, the key and its expiration time are distributed to the client in response to the request by the client. 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.”), [0082] (“The client will monitor the expiration time for each pay period and request a new set of keys from the caching server.”). A POSITA would have understood that a client device would request a new content encryption key (e.g., CK) before the expected expiration of the content encryption key currently being applied (e.g., FPK, GK, or CK) to “avoid service interruptions,” and therefore the client device would pre-fetch the next un-retrieved key within 100% (a “fixed percentage”) of the fixed periodic content encryption key expiration interval. APPLE-1003, ¶¶111-112; *supra*, Element [3]. Peterka thus discloses the additional features recited in claim 6 at least to the same extent described in the specification

of the '178 patent.² Cf. APPLE-1001, 11:58-62 (“the threshold is set as a fixed percentage of the fixed period content encryption key lifespan.”); *see also id.*, 9:39-43 (“In another embodiment, the threshold is set as a fixed percentage of the fixed period content encryption key lifespan (e.g., when the lifespan of the current content encryption key is 90% complete).”); APPLE-1003, ¶¶111-112.

Claim [7]

Peterka discloses that key distribution messages (e.g., ECMs/EMMs) include a “sequence number” to protect against “replay attacks.” APPLE-1004, [0121]; *supra*, Element [1.2]. The “sequence number” provides an identifier for the content encryption key contained in the key distribution message (*supra*, Element [1.2]), and as a result of the identifier value being assigned according to a “sequence,” a POSITA would have understood the “sequence number” to be predictable based on a predetermined known progression. APPLE-1003, ¶¶113-114. Likewise, binary implementations of the “parity bit” identifier follow a predetermined known progression that alternates between values of 0 and 1 to indicate whether the current or a new content key should be used. APPLE-1004, [0117]-[0121]; *supra*, Element

² Petitioner recognizes that IPR is not an appropriate forum for expressing deficiencies under 35 U.S.C. § 112, but reserves all rights to raise such defenses in other proceedings involving the '178 patent.

[1.2]; APPLE-1003, ¶114.

Claim [12]

Peterka discloses a client device that detects actual content encryption key rotation boundaries based on real-time in-band notifications, e.g., based on notifications such as a parity bit embedded directly in a packetized content stream. APPLE-1004, [0116] (“[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”), [0117] (“a predetermined bit” in an RTP packet “can be used 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”), [0118]-[0121], FIG. 14; APPLE-1003, ¶¶115-116.

Claim [13]

Peterka discloses a client device that detects content encryption key rotation boundaries based on notifications in an unencrypted header portion of the encrypted content files, e.g., in the header portion of an RTP packet. APPLE-1004, [0117] (“an extended header in which a predetermined value for that fixed field indicated that the content key for the payload portion of the packet has changed”), [0118] (“an RTP header extension could be utilized. In this way, the extended portion of the header could include at least the content key parity bit to indicate key changes”), [0120] (“padding bit in the RTP header could be utilized to indicate the key

change”); *generally id.*, [0116]-[0121]; APPLE-1003, ¶¶117-118.

Element [16.P]

To the extent the preamble is a limitation, Peterka discloses a client device (***computerized device***) operable as a client for handling secure distribution of content. APPLE-1004, [0038]-[0040] & FIG. 1 (“client computer 112”), [0041]-[0042] & FIG. 2 (describing “individual system elements” of devices including client 112); *supra*, Element [1.P], Section IV.A.1; APPLE-1003, ¶¶119-120.

Element [16.1]

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. APPLE-1004, [0041], FIG. 2; *generally id.*, [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.*; APPLE-1003, ¶121. The Peterka-Bocharov combination 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 [16.2]-[16.7]. *Supra*, Elements [1.1]-

[1.6]; Section IV.A.3(a); APPLE-1003, ¶121.

Element [16.2]

Supra, Element [1.1]; APPLE-1003, ¶122.

Element [16.3]

Supra, Element [1.2]; APPLE-1003, ¶123.

Element [16.4]

Supra, Element [1.3]; APPLE-1003, ¶124.

Element [16.5]

Supra, Element [1.4]; APPLE-1003, ¶125.

Element [16.6]

Supra, Element [1.5]; APPLE-1003, ¶126.

Element [16.7]

Supra, Element [1.6]; APPLE-1003, ¶127.

Claim [17]

Supra, Claim [2]; APPLE-1003, ¶¶128-129.

Claim [18]

Supra, Claim [4]; APPLE-1003, ¶¶130-131.

Claim [19]

Supra, Claim [7]; APPLE-1003, ¶¶132-133.

Claim [20]

Supra, Claim [12]; APPLE-1003, ¶¶134-135.

B. [Ground 1B]: Peterka in view of Bocharov, Peterka308, and Chen

1. Peterka308

Peterka308 describes techniques for encrypting and decrypting secured video segments for playback on a device. APPLE-1009, Abstract, FIGS. 2-6; APPLE-1003, ¶¶61-62. A DVR 104 “receives encrypted content from [a] content source 102.” APPLE-1009, [0021]. A “content 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.* For example, a “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.” *Id.*, [0024].

The actual CEK need not be “stored in the hard drive 202 or other media storage.” *Id.*, [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 id.*, [0026]-[0028].

2. Chen

Chen describes techniques for encrypting video on-demand (VOD) “material with a changing cryptographic key.” APPLE-1006, [0008]; *generally id.*, [0008]-[0011], [0027]-[0028], [0034]-[0038], [0035], FIGS. 2A-2D; APPLE-1003, ¶63. “The inventive 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.” APPLE-1006, [0034]. In particular, the four schemes are operable to define segments of encrypted video content based on a “fixed crypto period,” a “fixed number of packets,” a “fixed ‘marker’ count,” or a “random crypto period,” respectively. *Id.*; *generally id.*, [0037]-[0047].

3. Obviousness Based on Peterka-Bocharov-Peterka308-Chen Combination

(a) Overview of Combination

Peterka describes techniques for signaling key rotation based on a “parity bit” embedded in streaming media packets. APPLE-1004, [0117] (“a predetermined bit can be used 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”), [0118]-[0121]; APPLE-1003, ¶¶136-137; *see also id.*, ¶¶136-141. The parity bit is effective when a client holds keys for just two successive segments, e.g., the current segment and the next segment in a media stream. Indeed, when a client holds keys for just two

successive segments, a key change signaled by the parity bit would be sufficient to inform the client of which segment's key(s) to apply next since the client only holds key(s) for one other segment besides the current segment. *Id.* However, additional logic would be necessary for a client device to distinguish among keys for additional segments if the client simultaneously held keys for multiple segments beyond just the current segment and the immediately succeeding segment. *Id.* For example, Peterka's mention about the possibility of using a single bit (i.e., "if a single bit is used") suggests that longer, multi-bit parity values could alternatively be applied to uniquely distinguish among even more keys. APPLE-1004, [0117], FIG. 14.

To the extent Peterka does not expressly disclose a framework for client devices to select a next content encryption key to apply at key rotation boundaries in the scenario where a client has pre-fetched content encryption keys for multiple future segments, such techniques would have been obvious in view of Peterka and Peterka308. APPLE-1003, ¶138. As an initial matter, it would have been obvious to distribute content encryption keys (e.g., CKs, FPK, GK) to client devices for multiple future media segments at once. APPLE-1003, ¶138. Peterka expressly recognized the benefit of advanced distribution of keys for at least one future segment in order to reduce the "load" of client requests issued ahead of each segment boundary. APPLE-1004, [0080]. A POSITA would have appreciated that distributing keys for additional segments beyond just the current and next segments

would have beneficially furthered Peterka's goal of reducing the "load" at individual segment boundaries (e.g., since clients could pre-fetch keys for more segments at once and thereby reduce the frequency of each client's requests to fetch keys). APPLE-1004, [0080], [0106] ("pull key distribution model"); APPLE-1003, ¶138.

As discussed above, Peterka308 further disclosed techniques for correlating content encryption keys with information indicative of a location in a media stream where the content encryption key is to apply. *Supra*, Section IV.B.1; APPLE-1003, ¶139. Specifically, Peterka308 proposes to use 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. *Id.*, [0025]. In the context of a system like Peterka's (and a Peterka-Bocharov-Peterka308-Chen combination) that 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 of signaling to client devices which content encryption key (e.g., which CK, GK, or FPK) to apply at the start of each segment. APPLE-1003, ¶139. For example, a client that held keys for three future segments 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* APPLE-1004, [0117]-[0121]; APPLE-1009, [0025]. In comparison to Peterka's original parity bit solution, Peterka308's teaching would improve the resulting combination by allowing clients to more effectively manage keys obtained for segments multiple periods in advance. APPLE-1003, ¶139.

Furthermore, to the extent Peterka does not expressly disclose what criteria the content servers apply to subdivide media content into cryptographic "segments" (where each segment is encrypted by a different key), Chen discloses various criteria that were known to be applied for this task. *Supra*, Section IV.B.2. For example, according to Chen, a first option (referred to as a "fixed crypto period"), is to "[d]efine a crypto time interval and change the key each time the crypto time-interval passes." APPLE-1006, [0034], [0037]-[0038], FIG. 2A. A second option (referred to as "fixed number of packets"), is to "[d]etermine a number of content packets 'n' corresponding to a suitable time interval and change the cryptographic key every 'n' packets." *Id.*, [0034], [0039]-[0041], FIG. 2B. A third option (referred to as "fixed 'marker' count") is to "[u]se a suitable MPEG-II field type as a 'marker,' such as an I-frame header," and "change the cryptographic key every time 'n' markers have passed in the stream." *Id.*, [0034], [0042]-[0044], FIG. 2C. A POSITA would have found each of these options to be a predictable design alternative that would have been obvious to try in view of the limited number of methods available to specify locations in a media stream (e.g., Chen's four methods including fixed crypto period,

fixed number of packets, fixed market count, and random crypto period). APPLE-1003, ¶¶140-141. Each of Chen's first three options would also carry the obvious benefit of allowing the encoding device and the decoding device (e.g., clients in the resulting combination) to reference a common metric that could be readily obtained by each device so that the decoding device could 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. *Id.* 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. *Id.* 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. *Id.*

(b) Claim Element Analysis

Peterka, Bocharov, Peterka308, and Chen would have rendered obvious claims 7-9, 14-15, and 19, and the combination (*supra*, Section IV.B.3(a)) discloses each limitation of these claims as described below. APPLE-1003, ¶136.

Claim [7]

Peterka308 discloses content encryption key identifiers in the form of “time stamps,” where the time stamps are predictable based on a predetermined known progression (e.g., every “five minutes has elapsed”). APPLE-1009, [0024], [0029]; *supra*, Section IV.B.1. 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, or a fixed ‘marker’ count (e.g., fixed number of I-frames passed in an MPEG-II stream). APPLE-1006, [0034]-[0044], FIGS. 2A-2C; *supra*, Section IV.B.2.

For each of the reasons described in detail above (Section IV.B.3(a)), it would have been obvious to combine the teachings of 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. APPLE-1003, ¶¶142-143; *supra*, Section IV.B.3(a).

Claim [8]

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. APPLE-1006, [0034]-[0044], FIGS. 2A-2C; *supra*, Section IV.B.2. As described

above (Section IV.B.3(a)), it would have been obvious to combine the teachings of 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 count of frame markers passed in the media stream. APPLE-1003, ¶¶144-145; *supra*, Section IV.B.3(a). The frame marker count thus provides an identifier that is calculated as monotonically increasing sequential integer values (e.g., since the number of frames is an integer count that only increases over time) and is 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). APPLE-1003, ¶145.

Claim [9]

Peterka308 discloses content encryption key identifiers in the form of “time stamps.” APPLE-1009, [0024], [0029]; *supra*, Section IV.B.1. Chen further discloses techniques for segmenting media content based on criteria such as a “fixed crypto period” that “[d]efine[s] a crypto time interval” in which “the key [is changed] each time the crypto time-interval passes.” APPLE-1006, [0034], [0037]-[0038], FIG. 2A; *supra*, Section IV.B.2. As described above (Section IV.B.3(a)), it would have been obvious to combine the teachings of 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 or crypto time interval. APPLE-1003, ¶¶146-147; *supra*, Section IV.B.3(a). The “time stamp” as taught by Peterka³⁰⁸ and “crypto time interval” as taught by Chen both constitute “expected wall clock time[s] for applying the next content encryption key” since each references a playback time of the media stream to determine segment boundaries where the next content encryption key should apply. *Id.*; *cf.* APPLE-1001, 3:33-34 (“wall clock time *offset* from the beginning of the live stream”) (emphasis added). Each is also based on a “fixed periodic content encryption key expiration interval,” e.g., the fixed number of minutes taught by Peterka³⁰⁸ or the fixed periodic “crypto time interval” taught by Chen 756. APPLE-1009, [0024]-[0025]; APPLE-1006, [0034], [0037]-[0038], FIG. 2A; APPLE-1003, ¶147.

Claim [14]

For the reasons described below (Claim 15), it would have been obvious based on the teachings of Peterka, Bocharov, Peterka³⁰⁸, and Chen to detect content encryption key rotation boundaries based on notifications present in the content manifest file (e.g., based on timestamps of fragments in the client manifest). *Infra*, Claim 15; APPLE-1003, ¶¶148-150.

Additionally, Bocharov teaches that the manifest file can be an Internet Information Server (IIS) smooth streaming manifest file. APPLE-1005, [0036] (“smooth streaming system using ... Microsoft Internet Information Server (IIS)”),

[0038] (“using IIS, the administrator may publish a URL” and “[w]hether using MP4 or other container formats, the encoder may add additional boxes or information to the stream[] ... that contains a manifest”), [0010] (“IIS”), FIG. 2; *see also id.*, Abstract (“smooth streaming system”), [0014] (same), [0030] (describing a “typical client manifest” that includes “SmoothStreamingMedia” fields). A POSITA would have been motivated to apply Bocharov’s use of an IIS Smooth Streaming manifest file in the combination since Bocharov confirms that such a manifest file was suitable to carry fragment information of the types disclosed in Bocharov. APPLE-1003, ¶150. Further, a POSITA would have recognized benefits in 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. *Id.*

Claim [15]

Bocharov discloses that the client manifest contains, for each media fragment available in a stream, an “absolute timestamp” that indicates the location of the fragment within a larger content stream. APPLE-1005, [0030]-[0031].

```
<?xml version="1.0" encoding="utf-8"?>
<!--Created with Expression Encoder version 2.1.1205.0-->
<SmoothStreamingMedia MajorVersion="1" MinorVersion="0"
  Duration="6537916781"
  LookAheadFragmentCount="3" IsLive="TRUE">
  <StreamIndex Type="video" Subtype="WVC1" Chunks="327"
    Url="QualityLevels({bitrate})/Fragments(video={start time})">
    <QualityLevel Bitrate="1450000" FourCC="WVC1" Width="848"
      Height="480" CodecPrivateData=". . ." />
    <QualityLevel Bitrate="1050000" FourCC="WVC1" Width="592"
      Height="336" CodecPrivateData=". . ." />
    <c n="0" t="12345678" d="20000000" />
    <c n="1" t="32345678" d="20000000" />
    <c n="2" t="52345678" d="20000000" />
    <c n="3" t="72345678" d="20000000" />
  </StreamIndex>
</SmoothStreamingMedia>
```

APPLE-1005, [0030]; *see also id.*, [0014]-[0016].

According to Bocharov, “[e]ach fragment has a distinguished Uniform Resource Locator (URL) that allows the fragment to be identified and cached by both Internet cache servers and the client’s browser cache.” *Id.*, [0014]. The client uses the timestamps specified in the client manifest to “compose the fragment URL” and issues requests to retrieve particular fragments in the media stream. *Id.*, [0031], [0034] (“32- or 64-bit integer that indicates the absolute timestamp of the first sample”). To permit caching and retrieval of individual fragments, a POSITA would have understood that each fragment of media content in Bocharov corresponds to a different encrypted content file. APPLE-1003, ¶¶151-153. Likewise, a POSITA would have understood the file name of each fragment to be defined at least in part by the timestamps specified in the client manifest, particularly since URLs ordinarily

reference files by their filenames and the timestamps are used to form the fragment URLs. *Id.*

As previously described (Section IV.A.3(a)), each fragment in a combination based on Peterka and Bocharov would predictably correspond to a different encrypted program segment. APPLE-1003, ¶154. Alternatively, encrypted program segments could span multiple fragments or a single fragment could encompass multiple encrypted program segments. *Id.* Furthermore, based on the teachings Peterka³⁰⁸ and Chen, it would have been obvious to provide clients with encryption key identifiers that correlate encryption keys with the respective program segment to which the key applies—e.g., by providing key identifiers in the form of timestamps. *Supra*, Section IV.B.3(a), Claims 7, 9; APPLE-1003, ¶154. In this context, the fragment timestamps specified in the client manifest would predictably correspond to content encryption key rotation boundaries. *Id.* In particular, a client could readily compare the timestamp-based encryption key identifiers to the fragment timestamps to evaluate which key to apply to the fragment and detect key rotation boundaries. APPLE-1003, ¶154. In other words, the timestamps of streaming fragments, including those identified in the original manifest and those obtained thereafter, would provide in-band notifications embedded in the file names of the encrypted content file fragments from which clients would detect key rotation boundaries. *Id.* For example, if the timestamp filename of a fragment matched the

timestamp of a first pre-fetched content key (CK), the client would know that the start of the fragment corresponds to a key rotation boundary and that the first pre-fetched CK should be applied to decrypt the content of that fragment. *Id.*

A POSITA would have found it obvious to reference fragment timestamps in this manner in order to detect key rotation boundaries when streaming content packets like those disclosed in Peterka (e.g., RTP packets) are unavailable or when the stream of content packets is interrupted, as may occur at fragment boundaries. APPLE-1003, ¶155. For example, as disclosed in Bocharov, a client may download entire fragments at a time rather than stream pushed content packets from the server. APPLE-1005, [0015] (“unlike typical streaming where the server pushes data to the client, in this case the client pulls media fragments from the server”), [0007]-[0016], [0032] (“the server may send the fragment along with manifest information about the next few fragments”). A POSITA would have found the use of timestamps embedded in the filenames of fragments in the resulting combination to provide a useful and convenient solution for the client to obtain indications of key rotation boundaries so that appropriate keys could be applied to enable uninterrupted playback of media content across fragment and segment boundaries. APPLE-1003, ¶155.

Moreover, a POSITA would have understood that the timestamp necessary to detect key rotation boundaries could be obtained from any metadata associated with

a fragment that identifies the timestamp for the fragment. APPLE-1003, ¶156. The three pieces of metadata from which Bocharov indicates timestamps could be obtained include the initial client manifest, manifest information for subsequent fragments that accompanies requested fragments, and the filename of the requested fragments. APPLE-1005, [0030]-[0032]. Considering how each of these pieces of metadata are readily available to the client, each would have been obvious to try to extract the desired timestamp information from which the client could detect key rotation boundaries. A POSITA would have expected to achieve a high likelihood of success with each option, particularly since the timestamp information contained in each is equivalent. *Id.*

Claim [19]

Supra, Claim [7]; APPLE-1003, ¶¶157-158.

C. [Ground 1C]: Peterka in view of Bocharov and Balraj

1. Balraj

Balraj describes a secure content delivery service that allows users to obtain licensed media content for playback for a limited period of time. *See generally* APPLE-1007, [0046]-[0066], FIGS. 4-5; APPLE-1003, ¶64. Similar to Peterka’s solution, Balraj segments media content into cryptographic intervals referred to as “epochs.” *Id.* The system applies a respective presentation key to encrypt each epoch of the presentation, and the presentation keys are then distributed to user

devices 110 encrypted by respective service keys. APPLE-1006, [0053]-[0055]. A content server (CS) 300 is responsible for fetching encrypted presentation keys from a conditional access server (CAS) and distributing the encrypted presentation keys to clients before the start of each successive epoch. *Id.*, [0056]-[0066], FIGS. 4-5.

2. Obviousness Based on Peterka-Bocharov-Balraj Combination

(a) Claim Element Analysis

Peterka, Bocharov, and Balraj would have rendered obvious claim 5, and disclosed each limitation of claim 5 as described below. APPLE-1003, ¶159.

Claim [5]

Peterka discloses a “pull key distribution model” that “allows each client to actively request keys from the server.” APPLE-1004, [0106]; APPLE-1003, ¶¶160-163. Under the “pull key” model, “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.” APPLE-1004, [0106]; *see also id.*, [0104]-[0105], FIG. 8. To avoid service interruptions, the client must request and obtain the content key (and any secondary keys necessary to decrypt the content key) before the expiration of the content encryption key (e.g., CK, FPK, or GK) currently being used. *Id.*; *see also id.*, FIGS. 7A-7B; APPLE-1003, ¶161.

To the extent Peterka does not expressly disclose prefetching a next unretrieved key a fixed duration before the expected expiration of the content

encryption key currently being used, this feature would have been obvious to a POSITA based on the teachings of Balraj. APPLE-1003, ¶162. For example, Balraj defines a “Service License Prefetch time 430 for an Epoch” that represents “the earliest time at which the Service license will be available on the CAS for the associated Epoch (i.e. the earliest time at which the CS 300 can request encrypted presentation keys for that Epoch).” APPLE-1006, [0061]; *supra*, Section IV.C.1. According to Balraj, encrypted presentation keys (encrypted content encryption keys) for a next epoch must be requested after the period defined by the Service License prefetch time otherwise the keys for the epoch will not be available. APPLE-1006, [0061]-[0066], FIG. 5.

It would have been obvious to a POSITA to further modify the Peterka-Bocharov system in view of Balraj’s suggestion to implement an earliest prefetch time (e.g., a Service License prefetch time) that defines an earliest time at which keys for the next segment are made available for distribution from the server. APPLE-1003, ¶163. A POSITA would have appreciated that defining an earliest prefetch time would be beneficial for managing the load of client requests under the pull key distribution model by preventing or disincentivizing clients from requesting keys for a next segment too early (e.g., while the server continues to service requests from clients pertaining to the current or preceding segment). *Id.*; *see also* APPLE-1004, [0064], [0080], [0096]-[0097], [0126]-[0128], [0133] (Peterka’s repeated

concern with managing the server's "load"). Furthermore, under a pull-key distribution model in a system that defines an earliest prefetch time according to Balraj, it would have been obvious for clients to prefetch the next un-retrieved content key (and any other necessary secondary keys) at the earliest prefetch time (i.e., a "*fixed duration before the expected expiration of the content encryption key currently being used*") in order to maximize the time available for the server to return the next segment's keys. APPLE-1003, ¶163. In this way, the client would predictably increase the likelihood of obtaining the next segment's key(s) before the start of the next segment as needed to avoid service interruptions. *Id.*

D. [Ground 1D]: Peterka in view of Bocharov and Kelly

1. Kelly

Kelly, titled "Authentication System for Networked Computer Applications," discloses techniques for granting clients access to an application server using authenticated tokens. APPLE-1010, [0030] ("If the user 12 wishes to access the application server 32 again, the user 12 presents the encrypted combined token to the gatekeeper server 30 ..."); *see also id.*, [0042]-[0043], [0064]-[0066]; APPLE-1003, ¶65.

2. Obviousness Based on Peterka-Bocharov-Kelly Combination

(a) Claim Element Analysis

Peterka, Bocharov, and Kelly would have rendered obvious claim 10, and the

combination discloses each limitation of claim 10 as described below. APPLE-1003, ¶164.

Claim [10]

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. *See, e.g.*, APPLE-1004, [0066] (“The client delivers the secure object and a ticket with its own entitlement data (e.g. list of subscribed services, location, ability to pay for content, etc.) to the caching server. The caching server will examine the secure object and the ticket presented by the client to determine that the client selection from the secure object and the entitlement information from the ticket match the content access rules. If all rules are satisfied, the caching server will grant access to the requested content by delivering the content encryption key via the program key”), [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). Otherwise, access to the content will be denied.”), [0136] (“obtain a ticket which defines what type of content the client is entitled to obtain”), [0142]-[0144], FIGS. 19-21. Peterka explains that the client ticket or similar data record is “authenticated and encrypted so that the client cannot revise his own entitlements.”

Id., [0146], FIG. 23. The client can also employ a Unique Key to encrypt messages sent between the client and individual license servers (e.g., caching servers). *Id.*, [0095]. The client ticket thus provides a security “token” that allows the server to readily authenticate a client and determine whether to grant the client’s request for keys to access at least a portion of a streaming media content. APPLE-1003, ¶¶165-166.

To the extent Peterka does not expressly disclose that the client ticket or other token is encrypted with a symmetric key, these additional features would have been obvious to a POSITA. APPLE-1003, ¶167. Kelly, like Peterka, discloses techniques for encrypting security tokens—which was a conventional technique in the field to prevent tampering or unauthorized access to the content of the token. APPLE-1004, [0146]; APPLE-1010, [0028]-[0030] (“encrypts the authentication token with an encryption key”). Kelly also confirms that encrypting tokens with a “symmetric key” was a predictable option so long as the secret key is securely provided to all parties requiring it. APPLE-1010, [0042]-[0043] (“Symmetric encryption involves using a single key that is shared ...”); *see also id.*, [0064]-[0066].

A POSITA would have found it obvious to apply symmetric encryption to the security token (e.g., client ticket) in the Peterka-Bocharov-Kelly combination to achieve ordinary and predictable benefits associated with encryption, e.g., including to prevent tampering with the encrypted token such that it could not be successfully

decrypted and to prevent unauthorized access to the content of the token. APPLE-1003, ¶168. Moreover, “symmetric” encryption specifically would have been obvious to try given that it is one of the two foremost common encryption options (i.e., symmetric and asymmetric encryption) and does not require the creation of public-private key pairs (only one key needs to be maintained). *Id.*

E. [Ground 1E]: Peterka in view of Bocharov, Kelly, and Eisen

1. Eisen

Eisen, entitled “System and Method for Generating White-Box Implementations of Software Applications,” describes “[a] development suite for generating files, such as DRM-protected files, using white-box cryptography.” APPLE-1011, Abstract; *see also id.*, [0008]-[0009], [0026]-[0036], FIGS. 1-5; APPLE-1003, ¶66. Eisen discloses a user can “add[] white-box function calls to the appropriate white-box cryptography APIs to add functionality such as symmetric encryption[]” APPLE-1011, [0029].

2. Obviousness Based on Peterka-Bocharov-Kelly-Eisen Combination

(a) Claim Element Analysis

Peterka, Bocharov, Kelly, and Eisen would have rendered obvious claim 11, and the combination discloses each limitation of claim 11 as described below. APPLE-1003, ¶169.

Claim [11]

As discussed above, the teachings of Peterka, Bocharov, and Kelly disclose a token-based technique for securing and authenticating license server communications using a token encrypted with a symmetric key. *Supra*, Section IV.D (Claim 10); APPLE-1003, ¶¶170-171. Eisen further discloses that one type of symmetric encryption technique known before the Critical Date of the '178 patent was a “white-box”-based symmetric encryption. APPLE-1011, [0029] (“add[] white-box function calls to the appropriate white-box cryptography APIs to add functionality such as symmetric encryption[]”); *see also id.*, [0008]-[0009], [0026]-[0036], FIGS. 1-5; APPLE-1003, ¶171. The '178 patent does not identify any particular method of applying white-box encryption, nor does it identify any particular advantage associated with white-box encryption. *See* APPLE-1001, 10:15-17. Nonetheless, a POSITA would have found it obvious to apply Eisen’s teaching in combination with Peterka, Bocharov, and Kelly to use white-box symmetric encryption to encrypt the token (e.g., client ticket) that the client presents to the server in the resulting combination. APPLE-1003, ¶171. A POSITA would have been motivated to implement the combination to “provide security against white-box attackers.” APPLE-1011, [0026]. For example, white-box encryption techniques would beneficially provide protection against additional threats not covered by black-box cryptography techniques such as the attacker gaining control

of the execution environment and being able to view the encryption/decryption process. APPLE-1003, ¶ 171 (citing APPLE-1013, 1-2).

V. MANDATORY NOTICES UNDER 37 C.F.R. §42.8

A. Real Parties-In-Interest Under 37 C.F.R. §42.8(b)(1)

Petitioner Apple Inc. is the real party-in-interest. No other parties had access to or control over the present Petition, and no other parties funded the present Petition.

B. Related Matters Under 37 C.F.R. §42.8(b)(2)

Petitioner is not aware of any disclaimers, reexamination certificates or petitions for *inter partes* review for the '178 patent. Patent Owner has not asserted the '178 patent against Apple.

C. Lead And Back-Up Counsel Under 37 C.F.R. §42.8(b)(3)

Petitioner provides the following designation of counsel.

LEAD COUNSEL	BACK-UP COUNSEL
W. Karl Renner, Reg. No. 41,265 Fish & Richardson P.C. 3200 RBC Plaza 60 South Sixth Street Minneapolis, MN 55402 Tel: 202-783-5070 Fax: 877-769-7945 Email: IPR50095-0082IP1@fr.com	Thomas A. Rozylowicz, Reg. No. 50,620 Nicholas W. Stephens, Reg. No. 74,320 3200 RBC Plaza 60 South Sixth Street Minneapolis, MN 55402 Tel: 202-783-5070 Fax: 877-769-7945 Email: PTABInbound@fr.com

D. Service Information

Please address all correspondence and service to the address listed above.

Petitioner consents to electronic service by email at IPR50095-0082IP1@fr.com
(referencing No. 50095-0082IP1 and cc'ing PTABInbound@fr.com).

VI. PAYMENT OF FEES – 37 C.F.R. §42.103

Petitioner authorizes the Office to charge Deposit Account No. 06-1050 for the petition fee set in 37 C.F.R. §42.15(a) and for any other required fees.

VII. CONCLUSION

Petitioner respectfully requests institution of IPR and cancellation of all Challenged Claims based upon Grounds 1A-1E presented herein.

Respectfully submitted,

Dated: February 25, 2022

/W. Karl Renner/
W. Karl Renner, Reg. No. 41,265
Thomas A. Rozylowicz, Reg. No. 50,620
Nicholas W. Stephens, Reg. No. 74,320
Fish & Richardson P.C.
3200 RBC Plaza, 60 South Sixth Street
Minneapolis, MN 55402
T: 612-335-5070
F: 612-288-9696

(Control No. IPR2022-00618)

Attorneys for Petitioner

CERTIFICATION UNDER 37 CFR § 42.24

Under the provisions of 37 CFR § 42.24(d), the undersigned hereby certifies that the word count for the foregoing Petition for *Inter Partes* Review totals 13,925 words, which is less than the 14,000 allowed under 37 CFR § 42.24.

Dated: February 25, 2022

/W. Karl Renner/

W. Karl Renner, Reg. No. 41,265
Thomas A. Rozylowicz, Reg. No. 50,620
Nicholas W. Stephens, Reg. No. 74,320
Fish & Richardson P.C.
3200 RBC Plaza, 60 South Sixth Street
Minneapolis, MN 55402
T: 612-335-5070
F: 612-288-9696

(Control No. IPR2022-00618)

Attorneys for Petitioner

CERTIFICATE OF SERVICE

Pursuant to 37 CFR §§ 42.6(e)(4)(i) *et seq.* and 42.105(b), the undersigned certifies that on February 25, 2022, a complete and entire copy of this Petition for *Inter Partes* Review and all supporting exhibits were provided via Federal Express, to the Patent Owner by serving the correspondence address of record as follows:

ERICSSON INC.
6300 LEGACY DRIVE
M/S EVR 1-C-11
PLANO, TX 75024

/Crena Pacheco/
Crena Pacheco
Fish & Richardson P.C.
60 South Sixth Street, Suite 3200
Minneapolis, MN 55402
(617) 956-5938