

IPR2025-00003
Google LLC
v.
Ericsson AB
Patent 9,313,178

Patent Owner's Demonstratives



Presented January 27, 2026

Argument Roadmap

- ❖ Petition fails as to original claims
 - ❖ Peterka does not teach a single request yielding a single response with multiple CEKs
 - ❖ Petition's obviousness theory is contrary to Peterka's design purpose
 - ❖ Peterka does not teach or render obvious "content encryption key identifiers"
 - ❖ Peterka does not teach or render obvious the "detecting" [1D] and "issuing requests" [1E] limitations
- ❖ Contingent substitute claims are patentable
 - ❖ All regulatory and statutory requirements met
 - ❖ Petitioner fails to show unpatentability

Argument Roadmap

- ❖ Petition fails as to original claims
 - ❖ Peterka does not teach a single request yielding a single response with multiple CEKs
 - ❖ Petition's obviousness theory is contrary to Peterka's design purpose
 - ❖ Peterka does not teach or render obvious "content encryption key identifiers"
 - ❖ Peterka does not teach or render obvious the "detecting" [1D] and "issuing requests" [1E] limitations
- ❖ Contingent substitute claims are patentable
 - ❖ All regulatory and statutory requirements met
 - ❖ Petitioner fails to show unpatentability

Peterka does not teach a single request yielding a single response with multiple CEKs

Claims require one “request” and one “response” comprising *multiple* CEKs

Limitation 1[A]: “initiating *a* media playback request and receiving *a* playback request response”

Limitation 1[B]: “parsing content information from the playback request response, the content information including content encryption keys”

Cannot be met by:

- (1) A model providing only *one* key per request (Peterka’s pull model)
- (2) *Unrequested* keys (Peterka’s CK 0 and CK 1 in Fig. 7B)
- (3) Teachings that don’t disclose providing multiple keys in response to a *single* request (Peterka ¶ 80)

Peterka does not teach a single request yielding a single response with multiple CEKs

Petition relies only on Peterka's *pull* model

For obtaining keys, Peterka describes a push model (where a server keeps track of keys at each client and pushes keys as needed), a pull model (where a client keeps track of its keys and actively requests keys as needed), and a push-pull model (e.g., where a server only pushes keys to clients which confirmed receipt of previous key). Peterka, [0105]-[0115]. All these methods for *obtaining* keys are compatible with Peterka's signaling method informing clients when to *change* keys used for decryption. The Grounds herein rely on Peterka's *pull model*.

Every
element of claims 1 and 16 is satisfied by or obvious over Peterka's pull method.

Peterka does not teach a single request yielding a single response with multiple CEKs

Petition relies only on Peterka's pull model *for Limitations 1A & 1B*

Petition

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

Peterka discloses that each key (along with information about the key, *see infra* § IX.A.1.c) is “distributed to the client **in response** to the request by the client” in an EMM or ECM. Peterka, [0104], **FIG. 8**. The multiple EMMs/ECMs sent **in response** to a particular **media playback request** collectively constitute a **playback request response** received by a client, as claimed. Peterka, [0104]-[0106], [0121], [0122]-[0125], FIGS. 7B, 8; Wechselberger, ¶¶80-81.

Peterka

[0106] The pull key distribution model allows each client to **actively request** keys from the server. **In FIG. 8**, a flow chart **800** is illustrated for implementing **a pull** key distribution system.

Peterka does not teach a single request yielding a single response with multiple CEKs

Petition relies only on Peterka's pull model *for Limitations 1A & 1B*

~~As the Petition explained in detail, Peterka discloses both receiving keys in response to a request to access media content (Pet., 19-22, 27-28) and then “pulling” keys on an ongoing basis during playback (Pet., 34-36). In Peterka and in the Petition, reference to the “pull” model refers to *key* distribution (Peterka, [0105])—which, e.g., occurs *after* a request to access media content has been initiated and the first set of content keys have already been negotiated/received (Peterka, [0080]). Neither Peterka nor the Petition ever uses the word “pull” for *content*.~~

Peterka does not teach a single request yielding a single response with multiple CEKs

Peterka's pull model teaches providing only *one* key for *one* request

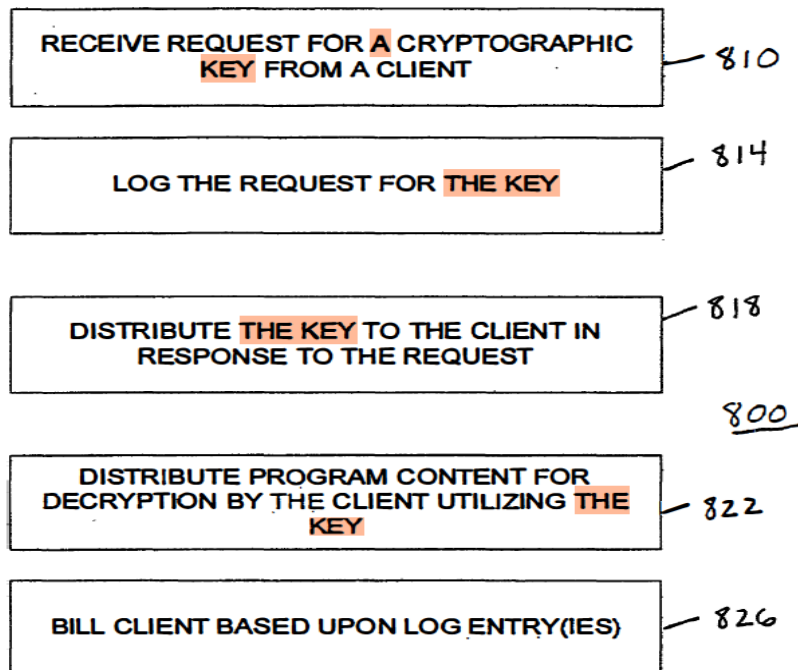


Fig. 8

[0110] Under a pull key distribution model, each client requests a new key and is individually responded to by the key distribution server.

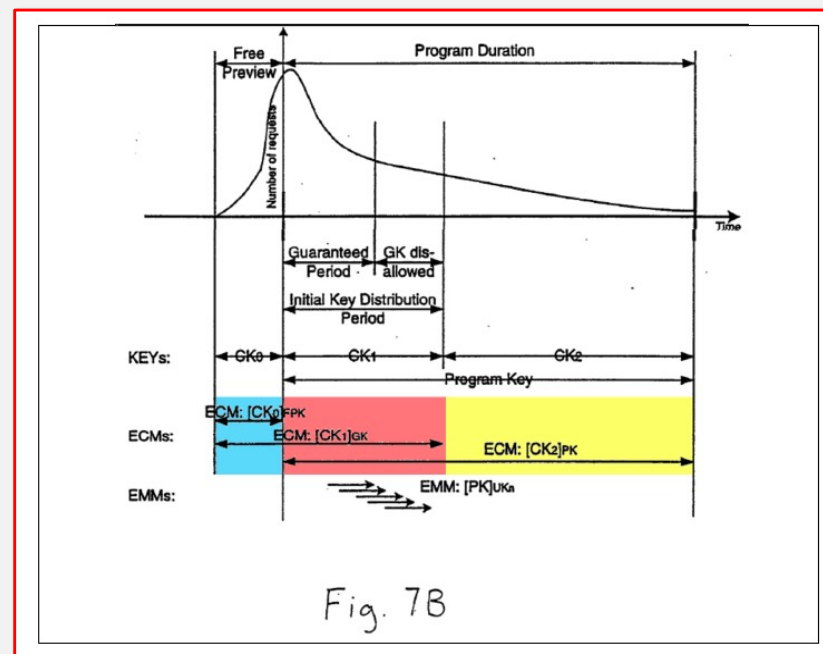
[0106] The pull key distribution model allows each client to actively request keys from the server. In FIG. 8, a flow chart 800 is illustrated for implementing a pull key distribution system. In block 810 a server receives a request for a cryptographic key from a client. The server logs the request for the key in block 814. For example, such a log entry could be made in a log such as a database maintained by the server. 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. In block 822, the program content is distributed for decryption by the client utilizing the cryptographic key that was requested. Finally, in block 826 the log entries can be utilized to bill the client based upon billing parameters of the billing arrangement.

54. Moreover, Peterka's teaching discloses that the Pay-By-Time distribution model operating in "pull" mode only provides *one* content key in response to each client request.

Peterka does not teach a single request yielding a single response with multiple CEKs

Petition relied heavily on CK0 and CK1 in Fig. 7B to meet Limitations 1A/1B

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



Peterka does not teach a single request yielding a single response with multiple CEKs

Peterka extensively teaches that CK0 and CK1 in Fig. 7B are *unrequested*

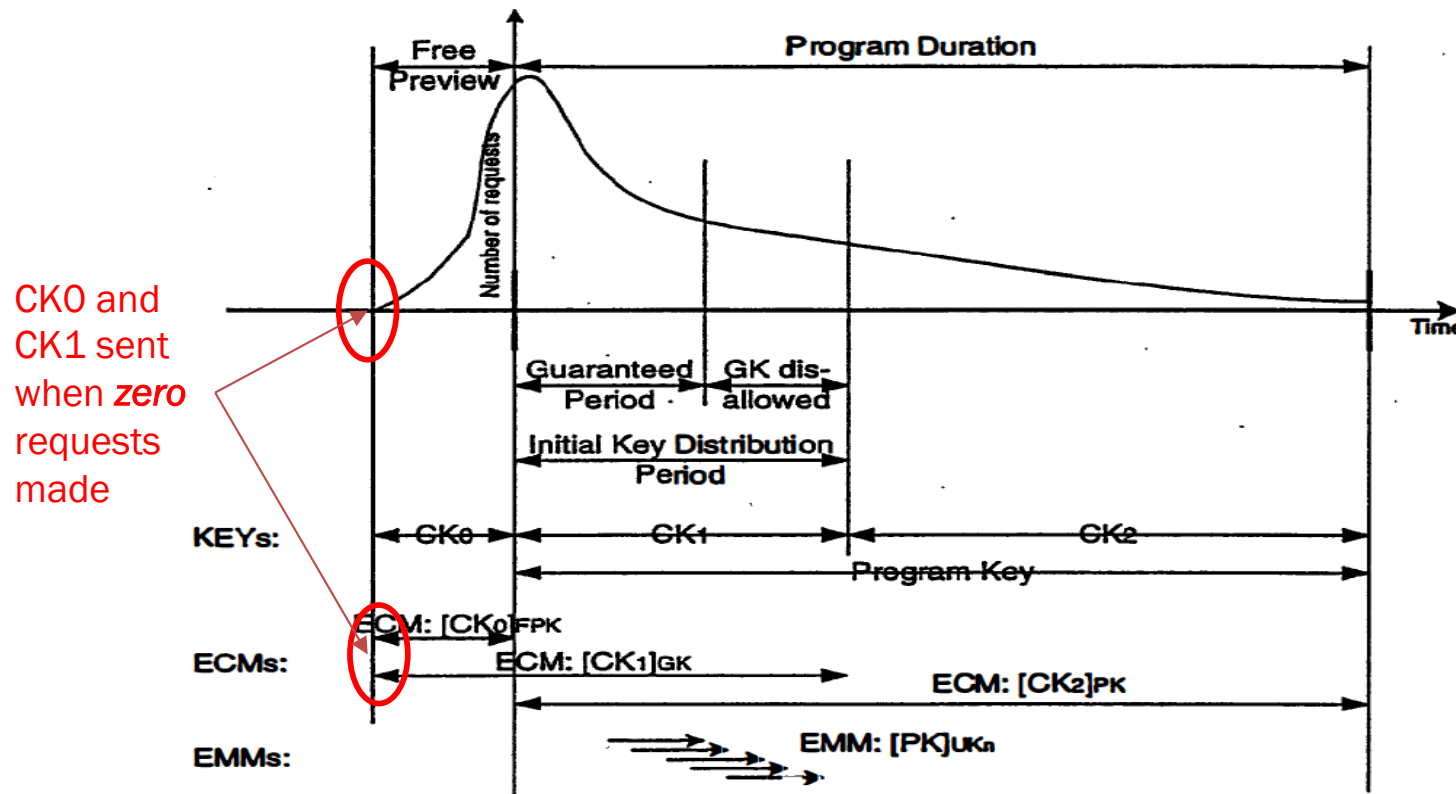


Fig. 7B

Peterka does not teach a single request yielding a single response with multiple CEKs

Peterka extensively teaches that CK0 and CK1 in Fig. 7B are *unrequested*

- “a free preview key [CK0] is provided ahead of time, e.g., during provisioning” ¶ 84
- “Clients will get the GK [group key, or CK1] *ahead* of program events sold on PPV [pay-per-view] or PBT [pay-by-time] basis, for instance during *provisioning*” ¶ 102
- “the group key is distributed ahead of time” ¶ 59

47. Because Peterka provides the content key for the free preview segment without request, this key (e.g., CK0 in Figs. 7A & 7B) cannot meet limitations [1B] and [16C] and thus cannot meet the challenged claims.

53. Because Peterka provides the content key for the “initial viewing period” segment *without request*, this key (e.g., CK1 in Figs. 7A & 7B) also cannot meet limitations [1B] and [16C] and thus cannot meet the challenged claims.

Peterka does not teach a single request yielding a single response with multiple CEKs

Peterka's paragraph 80 does not teach a single request or single response

[0080] For a quasi-PBT approach, a caching server may divide the content into pay segments and assign segment keys to them. All consumer computers would negotiate a key for each segment in order to keep track of how many segments they watched. This generates a large load on the caching server on the segment boundaries. This can be mitigated with a key management approach where each rabbit is given keys to the current as well as the next segment. This will give the caching server enough time to distribute the next segment keys during the current segment.

60. The Petition's reliance on the statement in Peterka that "each rabbit³ is given keys to the current as well as the next segment," Ex. 1005 ¶80, likewise does not support the Petition's assertions. In that passage, Peterka does not teach providing content keys to two segments *in response to a single request*, as the challenged claims require and the Petition contends.

Peterka does not teach a single request yielding a single response with multiple CEKs

Paragraph 80 does not change that a “pull” request yields only *one* key

[0080] For a quasi-PBT approach, a caching server may divide the content into pay segments and assign segment keys to them. All consumer computers would negotiate a key for each segment in order to keep track of how many segments they watched. This generates a large load on the caching server on the segment boundaries. This can be mitigated with a key management approach where each rabbit is given keys to the current as well as the next segment. This will give the caching server enough time to distribute the next segment keys during the current segment.

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

Peterka discloses that each key (along with information about the key, *see infra* § IX.A.1.c) is “distributed to the client *in response* to the request by the client” in an EMM or ECM. Peterka, [0104], FIG. 8. The multiple EMMs/ECMs sent *in response* to a particular *media playback request* collectively constitute a *playback request response* received by a client, as claimed. Peterka, [0104]-[0106], [0121], [0122]-[0125], FIGS. 7B, 8; Wechselberger, ¶¶80-81.

Petition

Argument Roadmap

- ❖ Petition fails as to original claims
 - ❖ Peterka does not teach a single request yielding a single response with multiple CEKs
 - ❖ Petition's obviousness theory is contrary to Peterka's design purpose
 - ❖ Peterka does not teach or render obvious "content encryption key identifiers"
 - ❖ Peterka does not teach or render obvious the "detecting" [1D] and "issuing requests" [1E] limitations
- ❖ Contingent substitute claims are patentable
 - ❖ All regulatory and statutory requirements met
 - ❖ Petitioner fails to show unpatentability

Petition's Obviousness Theory As to Limitations 1A/1B Fails

Figure 7B cannot support obviousness as to what it does not teach, i.e., providing multiple CEKs in response to *one* request

Peterka's Paragraph 104 teaches an entirely different concept: sending *one* key to one client in a package containing keys not directed to that client

Modifying Peterka to provide multiple CEKs in response to one pay by time pull request in a multicast distribution would defeat Peterka's design purpose by giving clients access to paid content segments they *did not purchase*

Argument Roadmap

- ❖ Petition fails as to original claims
 - ❖ Peterka does not teach a single request yielding a single response with multiple CEKs
 - ❖ Petition's obviousness theory is contrary to Peterka's design purpose
 - ❖ Peterka does not teach or render obvious "content encryption key identifiers"
 - ❖ Peterka does not teach or render obvious the "detecting" [1D] and "issuing requests" [1E] limitations
- ❖ Contingent substitute claims are patentable
 - ❖ All regulatory and statutory requirements met
 - ❖ Petitioner fails to show unpatentability

Peterka does not teach “content encryption *key identifiers*”

Specification defines “content encryption key identifiers”

“Each content encryption key in the history is ***assigned*** a ***unique*** identifier which is ***referred to herein*** as the content encryption key identifier.”

Ex. 1001 at 3:38-40

“The patentee’s use of the phrases ‘as used herein’ and ‘refer to’ conveys an intent for sentence 5 to be ***definitional.***”

ParkerVision, Inc. v. Vidal, 88 F.4th 969, 976 (Fed. Cir. 2003)

Peterka does not teach “content encryption *key identifiers*”

Petition relies on *message* identifiers, not *key* identifiers

Peterka discloses that each ECM/EMM includes “a *sequence number* or a *time stamp* to protect against replay attacks.” Peterka, [0121]. Because the sequence number/timestamp is unique to each ECM/EMM (to prevent replay attacks), and each ECM/EMM carries a different content encryption key (*supra* § IX.A.1.b), the sequence number/timestamp each constitutes a unique *content encryption key identifier*.

[0121] In multicasting program content such as audio and visual material, entitlement management messages and entitlement control messages can be sent from a server to client computer. 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.

Peterka’s message identifiers are not *assigned* to a CEK, and are not *unique* to a CEK

Peterka does not render “content encryption *key identifiers*” obvious

Petition lacks a *motivation* for modifying Peterka to add key identifiers

Peterka discloses that each ECM/EMM includes “a *sequence number* or a *time stamp* to protect against replay attacks.” Peterka, [0121]. Because the sequence number/timestamp is unique to each ECM/EMM (to prevent replay attacks), and each ECM/EMM carries a different content encryption key (*supra* § IX.A.1.b), the sequence number/timestamp each constitutes a unique *content encryption key identifier*.

Put another way, according to the Petition, and Mr. Wechselberger, Peterka’s system *already* allowed for differentiation between keys, albeit using *message* identifiers that do not meet the recited “*key identifiers*” of the challenged claims. Pet. 24, Ex. 1003 ¶89. Thus, the Petition has not identified any benefit or improvement to Peterka’s system from adding key identifiers that would have motivated an ordinary artisan to do so, and I disagree that the ordinary artisan would have found such a modification obvious.

Petition’s Ground 5 motivation rests on false premise: Peterka does not provide multiple CEKs to one client *for multiple* segments, thus no need for unique key identifiers

Petition merely shows what a POSA *could have* done, not a *reason why* she *would have*

Argument Roadmap

- ❖ Petition fails as to original claims
 - ❖ Peterka does not teach a single request yielding a single response with multiple CEKs
 - ❖ Petition's obviousness theory is contrary to Peterka's design purpose
 - ❖ Peterka does not teach or render obvious "content encryption key identifiers"
 - ❖ Peterka does not teach or render obvious the "detecting" [1D] and "issuing requests" [1E] limitations
- ❖ Contingent substitute claims are patentable
 - ❖ All regulatory and statutory requirements met
 - ❖ Petitioner fails to show unpatentability

Peterka's pull method cannot meet “detecting” in [1D]

Specification defines “detecting” limitation to require “check[ing] to see if a key change is signaled”

FIG. 3 is a flow chart 300 describing a process for detecting the rotation of content encryption keys and retrieving updated content encryption keys.

Once the new segment is retrieved, processing proceeds to step 308 where the client 110 checks to see if a key change is signaled, i.e., a key change was requested outside of any fixed duration lifespan. This can occur for security reasons as mentioned above.

In step 310, the client 110 checks the expiration of current content key to see if a new key should be applied. There are two cases—normal period-based expiration, and an explicit key change notification received in step 308 that applies to the current segment being decrypted.

In step 316, the client 110 checks to see if the current content encryption key is going to expire in the near future or if a content encryption key change request for a future segment was detected in step 308.

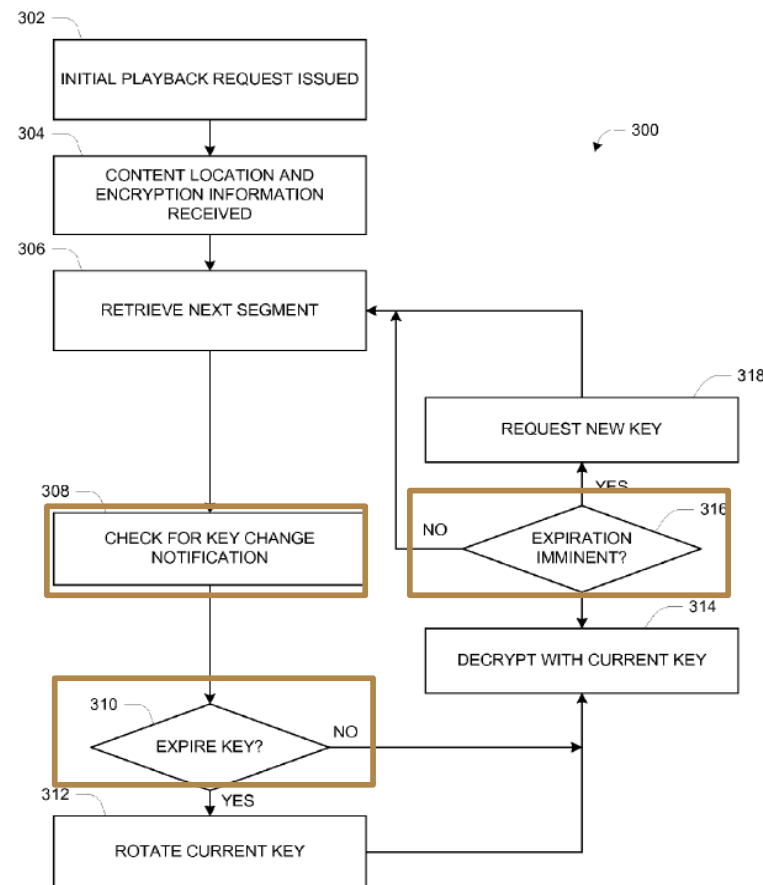


Fig. 3

Peterka's pull method cannot meet "detecting" in [1D]

Petition relies on Peterka's "pull" method, which does *not* check for a key change signal

Every
element of claims 1 and 16 is satisfied by or obvious over Peterka's pull method.

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. Alternatively, the push model migrates the responsibility to the server which keeps track of active clients and distributes new keys to them before the current keys expire.

[0106] The pull key distribution model allows each client to actively request keys from the server. In FIG. 8, a flow chart 800 is illustrated for implementing a pull key distribution system. In block 810 a server receives a request for a cryptographic key from a client. The server logs the request for the key in block 814. For example, such a log entry could be made in a log such as a database maintained by the server. 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.

Peterka's pull method cannot meet "detecting" in [1D]

Peterka's change signal notification requires the server to monitor key status, and is not compatible with Peterka's pull method

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. Alternatively, the push model migrates the responsibility to the server which keeps track of active clients and distributes new keys to them before the current keys expire.

[0106] The pull key distribution model allows each client to actively request keys from the server. In FIG. 8, a flow chart 800 is illustrated for implementing a pull key distribution system. In block 810 a server receives a request for a cryptographic key from a client. The server logs the request for the key in block 814. For example, such a log entry could be made in a log such as a database maintained by the server. 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.

[0116] As part of the key distribution system, the content key is utilized for decrypting the program content throughout the course of the distribution of the program content. Thus, 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.

Peterka's change signal cannot meet “issuing requests” in [1E]

Under Peterka's change signal notification, clients “*have been provided*” the next key, and clients do not issue requests for that key as required by limitation [1E]

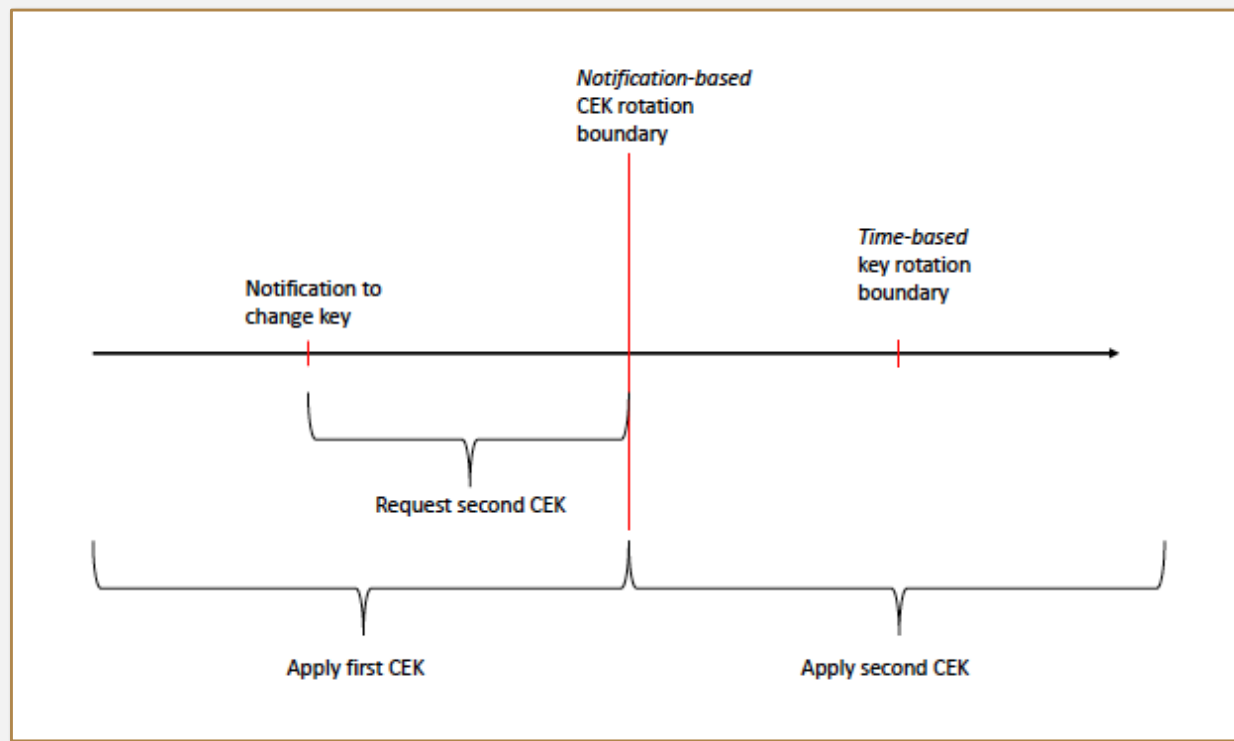
[0116] As part of the key distribution system, the content key is utilized for decrypting the program content throughout the course of the distribution of the program content. Thus, when a new content key is implemented, the implementation will be signaled to the clients so that the clients can begin utilizing the new content key with which they have been provided.

Argument Roadmap

- ❖ Petition fails as to original claims
 - ❖ Peterka does not teach a single request yielding a single response with multiple CEKs
 - ❖ Petition's obviousness theory is contrary to Peterka's design purpose
 - ❖ Peterka does not teach or render obvious "content encryption key identifiers"
 - ❖ Peterka does not teach or render obvious the "detecting" [1D] and "issuing requests" [1E] limitations
- ❖ Contingent substitute claims are patentable
 - ❖ All regulatory and statutory requirements met
 - ❖ Petitioner fails to show unpatentability

Revised amended claims do not broaden scope

Both original and revised claims require requesting a second CEK before the relevant boundary and applying that second CEK after that boundary



Revised amended claims do not introduce new matter

Written description supports notification-based early key expiration

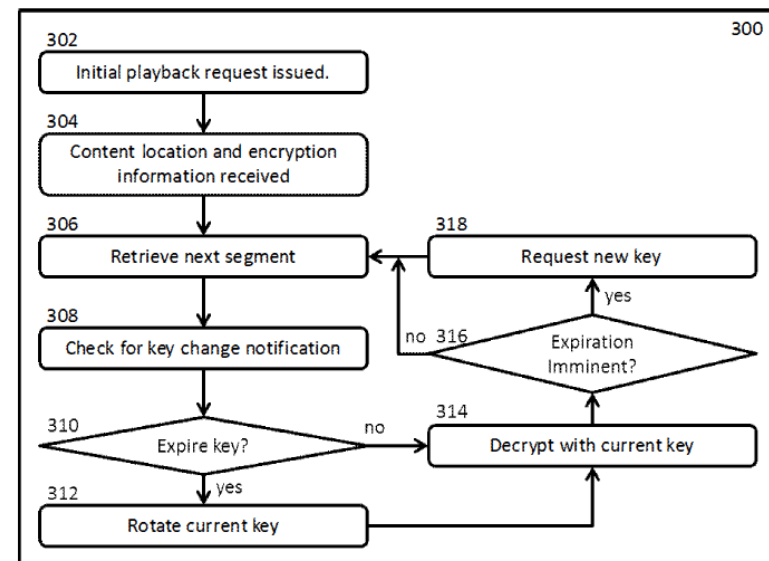
Once the new segment is retrieved, processing proceeds to step 308 where the client 110 checks to see **if an unscheduled key change is signaled, i.e., a key change was requested outside of any fixed duration lifespan**. In one embodiment, the packager 104 notifies the client 110 of the key change by prepending a header to the encrypted content which contains a flag that describes **the expiration of the previous key**.

In one embodiment, the workflow manager may initiate content encryption key rotation **at any time, outside of the fixed duration lifespan of the existing key**.

The content encryption key may be an unsolicited content encryption key pushed by the WFM 102 **as necessitated by security policies** (e.g., the current content encryption key **was compromised** or the current content encryption was found to be **not sufficiently random**).

In one embodiment, the expiration of the new content encryption key is set to the expiration time of the previous content encryption key, i.e., **if the new content encryption key was applied when the previous content encryption key had 10 minutes left in its lifespan**, then the lifespan of the new content encryption key would be set to 10 minutes.

FIG. 3



Argument Roadmap

- ❖ Petition fails as to original claims
 - ❖ Peterka does not teach a single request yielding a single response with multiple CEKs
 - ❖ Petition's obviousness theory is contrary to Peterka's design purpose
 - ❖ Peterka does not teach or render obvious "content encryption key identifiers"
 - ❖ Peterka does not teach or render obvious the "detecting" [1D] and "issuing requests" [1E] limitations
- ❖ Contingent substitute claims are patentable
 - ❖ All regulatory and statutory requirements met
 - ❖ Petitioner fails to show unpatentability

Petitioner fails to show unpatentability

Limitation [21.2]: Peterka teaches *message* identifiers, not *key* identifiers, and Petitioner fails to provide a motivation for obviousness

Limitation [21.5]: Neither Peterka nor Traw teaches a client requesting a new key “*in response to*” a key change notification. In Peterka, the new key has already been provided *without request*. Traw teaches the same.

Like Peterka, Traw expressly teaches that a notification instructs the client to change to a key that the client has already been provided. Ex. 1041 at Abstract (“a device manufacturer embeds the keys in content protection implementations for the devices, and to renew the content protection implementations in devices after the devices are distributed, in response to at least a security compromise”); 5:7-65 (“[T]he device manufacturer embeds the keys in manufactured devices ... [and] cause[s] the periodic renewal of the devices’ content protection implementations, which may include new keys in the new content protection implementation installed in the devices.”).

Petitioner fails to show unpatentability

Claim 23: Petitioner relies on an obvious to try theory without credibly showing that the limitation at issue was one of a finite and obvious set of options

Claim 30: Neither Peterka nor Traw teaches assigning a second key to the same *segment*