(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 September 2012 (07.09.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/118604 A1

(51) International Patent Classification:
*H04N 21/4405* (2011.01) *H04N 21/835* (2011.01)
*H04N 21/4408* (2011.01) *H04N 21/6334* (2011.01)
*H04N 21/433* (2011.01)

(21) International Application Number:
PCT/US2012/024400

(22) International Filing Date:
9 February 2012 (09.02.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/447,594 28 February 2011 (28.02.2011) US
13/331,266 20 December 2011 (20.12.2011) US

(71) Applicant *(for all designated States except US)*: **SANDISK TECHNOLOGIES INC.** [US/US]; Two Legacy Town Center, 6900 North Dallas Parkway, Plano, TX 75024 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **HAHN, Judah, Gamliel** [US/IL]; 24 Eretz Yemini Street, 90627 Ofra (IL). **SHMUEL, Avraham** [IL/IL]; Habrosh Tzafon Street, PO Box 624, 44935 Sde Warburg (IL).

(74) Agent: **GENIN, Kent, E.**; Brinks Hofer Gilson & Lione, P.O. Box 10087, Chicago, IL 60610 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

*[Continued on next page]*

(54) Title: METHOD AND APPARATUS FOR PROTECTING CACHED STREAMS

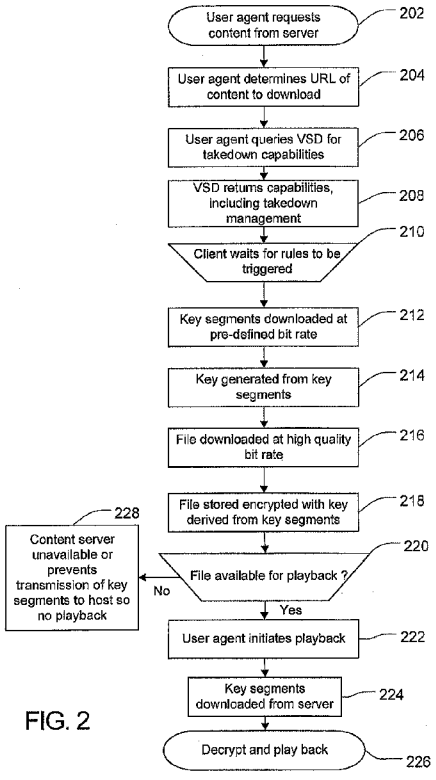


FIG. 2

(57) Abstract: A system and method for protecting cached streamed data is disclosed. The method may include the steps of generating an encryption key from the streamed data itself, encrypting the streamed data stored in the storage device and requesting the portion of the streamed data from the content server again when later playback is desired so as to allow the content server to enforce access limitations or takedown policies relating to the streamed data. The method may also include procedures for handling key generation over reliable or unreliable protocols.

WO 2012/118604 A1

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Published**:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

# METHOD AND APPARATUS FOR PROTECTING CACHED STREAMS

## CROSS-REFERENCE TO RELATED APPLICATIONS

**[0001]** This application claims the benefit of U.S. Provisional Application Serial No. 61/447,594, filed February 28, 2011, the entirety of which is hereby incorporated herein by reference.

## BACKGROUND

**[0002]** Numerous mobile devices are currently available that allow a user to send and receive data and stay connected with work, family and friends with relatively few geographical restrictions on the user. The widespread availability of sophisticated mobile devices, such as smart phones or mobile computing devices, coupled with the increased availability of high quality wireless networks, has led to a demand for more access to online content by mobile device users. Online content in the form of streamed data may now be routinely accessed and consumed by mobile device users over various wireless networks. To avoid potential variations in connection quality, or even loss of connectivity, during playback, some mobile devices may cache streamed data locally for later playback.

**[0003]** While caching of data can improve user experience, the data may have restrictions associated with it that cannot be readily enforced if it is cached. For example, if the online content is subject to Digital Millennium Copyright Act (DMCA) restrictions in the United States, the content provider may be required to remove access to online content upon notice from a content owner of potential copyright infringement. Alternatively, content may be subject to geographical licensing restrictions, in which the content is freely available only in certain regions and cannot be consumed outside of these regions. When content is streamed online for immediate consumption, the server can easily restrict service by simply not streaming the content if it is restricted or removed. If the content is cached, it becomes more difficult to impose these restrictions. One way to resolve

this issue is via digital rights management (DRM) software. However, DRM requires infrastructure changes on the content provider server and typically carries a per-copy license fee to the DRM provider. This may be impractical for user-generated content services such as YOUTUBE and FLICKR.

## SUMMARY

**[0004]** In order to address the problem of handling enforcement of restrictions on, or removal of access to, streamed content in a system that caches streamed content, a system and method for protecting cached streams is disclosed.

**[0005]** According to one aspect, a method for protecting streamed content that has been stored locally on a storage device associated with a host is disclosed. The method may include, in a processor of the host, requesting streamed content from a content server, and creating an encryption key from a portion of the streamed content. The portion of the streamed content used to create the encryption key may be downloaded at a different bit rate than the bit rate at which the streamed content is downloaded for storage on the storage device. The processor may determine if the protocol between the content server and a user agent on the host is unreliable and, if so, repeatedly retrieve portions of the streamed content sufficient to create the encryption key until enough of the portions retrieved have been reliably retrieved to form the key. The streamed content is then encrypted with the key and stored in the storage device with an encrypted version of the key, where the encrypted version of the key is encrypted with the retrieved portion of the streamed content and also stored in the storage device. The unencrypted key is then discarded. In alternative embodiments, one or more overlay keys may be used to generate a combined encryption key with the encryption key generated from a portion of the streamed content.

**[0006]** In another aspect, a method of decrypting streamed content is disclosed where, in a processor of a host device, upon receiving a request to playback encrypted streamed content, an encrypted key is retrieved from a storage device associated with the host and the encrypted key is decrypted by requesting an

unencrypted version of the encrypted streamed content from a content server. Portions of the unencrypted version of the streamed content are received over an unreliable protocol and used to decrypt the encrypted key. Upon decryption of the encrypted key, the decrypted key is used to decrypt the encrypted streamed content on the storage device and play back an unencrypted version of the encrypted streamed content from the storage device. In alternative embodiments, the storage device or the host may manage encryption and decryption processes.

**[0007]** According to another aspect, a host is disclosed having a processor and a user agent, the user agent configured to communicate with a content server containing streamed content and the processor configured to encrypt and decrypt streamed content according to the methods disclosed above. In yet another aspect, a storage device is disclosed for use in a host that is configured to communicate with a content server to obtain streamed data, and store the streamed data on the storage device, where the storage device is configured to encrypt and decrypt the streamed data in accordance with the methods disclosed above.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0008]** FIG. 1 is a block diagram of a system that may implement aspects of the invention.

**[0009]** FIG. 2 is a flow diagram illustrating one embodiment of a method for protecting a streamed data cached locally on a storage device.

**[0010]** FIG. 3 is a flow diagram illustrating a method of deriving an encryption key from streamed content for use in the method of FIG 2.

**[0011]** FIG. 4 is a flow diagram illustrating a method of decrypting the encrypted key formed in the process of FIG. 3

**[0012]** FIG. 5 illustrates an alternative embodiment of the system of FIG. 1.

**[0013]** FIG. 6 is a flow diagram illustrating an alternative implementation of the method of FIG. 2 for use in the system of FIG. 5.

**[0014]** FIG. 7 is a flow diagram illustrating an alternative implementation of the method of FIG. 3 incorporating an overlay key in addition to a key derived from the streamed content.

**[0015]** FIG. 8 is a flow diagram illustrating an alternative implementation of the method of FIG. 4.

**[0016]** FIG. 9 is a flow diagram illustrating an alternative implementation of the method of FIG. 8.

## DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EMBODIMENTS

**[0017]** A suitable system for implementing methods for protecting streamed data is illustrated in FIG. 1. In the system, a content server or collection of servers 102 in a network 106 may be in communication over wired or wireless communications links 103 with a host 104. The host may be a mobile device such as any of a number of data handling devices including, but not limited to a tablet computer, mobile phone, personal digital assistant, home network router, or a personal computer. A communication channel between the content server or servers 102 and the host 104 may be via any of a number of available wired or wireless interfaces. The host 104 may include an integrated or removable storage device 108. The storage device 108 may be a non-volatile memory. One suitable type of a non-volatile memory is a flash memory card. The host 104 may alternatively, or additionally, be coupled to an external storage device (e.g. flash memory card or USB flash drive coupled via a memory card slot or USB interface) suitable for holding cached streamed content. The term streamed data (or streamed content) is generally used herein to refer to data that is capable of being streamed for consumption at a real-time streaming bit rate, but that may be transferred, for example pre-fetched from a content server and cached in a storage device, at a bit rate other than the bit rate at which it would be transmitted for real-time consumption.

**[0018]** The host 104 includes a processor 110 for executing a client application 112 that may include one or more virtual storage devices 114 and a caching engine

116. A user agent 118, such as a video playback application, may also be stored on the host. The user agent 118 may be a browser or other user interface that allows access to streamed content with the host 104. Any number of available types of browsers, such as Internet Explorer, Safari, YouTube handset application, or Chrome may be used in this context. The caching engine 116 may be software for managing the timing and handling of data a user wishes to download. The sources of data desired by a user may be stored in a queue manager 120 in the caching engine 116 for managing and prioritizing user requests for downloading data from the one or more sources. The download of data from the source identified in the queue manager 120, such as a content server 102, would be initiated based on the rules for that source set out in an integrated or remote policies database 122. For example, rules may be associated with a streamed video of a television episode to be cached in the storage device 108. The rules in the policy database 122 pertaining to the source may be to only download and store the latest episode after a certain date when the host 104 is otherwise idle, or to download and store the episode when the host 104 is in communication with a content streaming server 102 over an advantageous type of network connection (e.g. cost or bandwidth), or so on.

**[0019]** The virtual storage device (VSD) 114 may be implemented as an abstraction layer application programming interface (API) representing the storage device 108 and acting as a driver for the storage device 108. The virtual storage device 114 may also be implemented with additional capabilities such as auto-backup, re-encryption or other functionality. The client application 112 may be already resident on the host 104, or downloaded in whole or in part from a remote source prior to or concurrently with downloading of streamed content. In yet other alternative embodiments, some or all of the client application 112 may be retrieved from the storage device 108 that may be coupled with the host 104.

**[0020]** In order to protect the streamed data and allow content servers to control access of data that is downloaded by the host and stored locally on the storage device associated with the host, disclosed herein is a mechanism for

encrypting streamed data stored on the storage device 108 of the host 104 such that future playback of the streamed content that is cached on the host 104 is dependent on a key that must be recreated by accessing the streamed content again from the content server 102 to recreate the key that was used to encrypt the streamed content when it was first cached. Thus, by not storing the key generated at the host 104 and requiring that the host contact the content server 102 to stream some of the cached streamed content and recreate the key, the content server 102 can control the ability of the host to play back cached streamed data even though the streamed data is already cached locally on the storage device 108 associated with the host 104. As described in greater detail below, in one implementation the technique for deriving a key from the streamed data allows for a certain number of erroneous bits that may be retrieved if the protocol used to transfer the streamed data is an unreliable data protocol.

**[0021]** In one embodiment, as best seen in FIG. 2, a process of downloading and encrypting the downloaded streamed content is presented. When the user agent 118 first requests content from a content server (at 202) the user agent 118 determines the universal resource locator (URL) of the content to be downloaded. This may be accomplished for example by an internal algorithm on the client application 112 in the host 104 or, as in a YouTube request, via a communication directly with the content server (at 204). In one particular implementation, the user agent 118 may then query the virtual storage device 114 on the host 104 for its takedown capabilities, wherein the "takedown" capability refers to the ability of the virtual storage device 114 for the host 104 to prevent playback of data restricted by a content server (at 206). The takedown capability may include the ability of the virtual storage device 114 to completely eliminate all local copies of content in response to a takedown command. Also, in one alternative embodiment, the host 104 may contain more than one virtual storage device 114, each with different capabilities. For example, two different virtual storage devices may be present, one with takedown capabilities and one without, where the user agent may select the virtual storage device with the takedown capabilities.

[0022] The user agent query may return a flag or a file from the virtual storage device 114 simply noting that the capability exists or providing further details on the capability. In addition, other capabilities of the virtual storage device 114, such as auto backup or other functions, may also be returned in response to the user agent 118 query (at 208). At this point, the caching engine 116 of the client 112 will compare the user request for content against the rules for triggering download to determine what to download and when to download (at 210). The caching engine 116 of the client 112 may use rules from the policy database 122 to control downloads. For example, rules concerning when to download based on time of day, bandwidth available, power level of the mobile device, or any of a number of other parameters utilized by the caching engine 116. The caching engine 116 handles system-generated downloads and user-initiated content downloads from content servers 102 in various remote locations.

[0023] Once the request for content has been processed and the caching engine 116, pursuant to the rules being implemented for downloading, have been triggered, key segments are downloaded from the content server 102 at a predefined bit rate (at 212). The predefined bit rate may be the lowest available bit rate known to be available to the host. The lowest bit rate information may be information provided to the host by the content server. The key segments are fragments of streaming data retrieved from the server 102. Thus, the key fragments may be portions of the content that is to be cached, for example video content. In one embodiment, a small portion of the beginning of the stream is retrieved at a low bit rate and that segment is hashed to return an encryption key. Although the encryption key may be generated from a single portion of the beginning of the stream of streamed data if a reliable protocol, such a transmission control protocol (TCP)-based protocol, is being used, and the subsequent downloaded data may be simply encrypted with that key from the single segment, it is expected that an unreliable data protocol may be involved and thus the modification such as shown in FIG. 2 may be required. One example of an unreliable protocol is User Datagram Protocol (UDP).

**[0024]** Streaming is inherently unreliable, and it is possible that the same stream retrieved multiple times from a content server 102 may result in a different set of bits. Therefore, in a second embodiment of this invention, multiple segments of the key are derived from different portions of the stream. Each segment may be used to decrypt a partial shared secret, which is used to form the final key. This is illustrated in FIG. 3.

**[0025]** Returning again to FIG. 2, after downloading the key segments at a predefined bit rate, a key is generated from the key segments and the file is downloaded at a high quality bit rate, and subsequently the file is stored encrypted based on the key derived from the key segments (at 214-218). A high quality bit rate is defined herein as a bit rate suitable for viewing data at the best rate the destination device (host) can manage over a Wi-Fi connection. For example, if a host can process and display video at bit rates of 80 kbps through 2.5 Mbps, the high quality bit rate would be 2.5 Mbps. Although the streamed content (also referred to herein as a file) is encrypted and stored in the storage device associated with the host while the unencrypted key is discarded from the host 104. Thus, the file is available for playback but requires generation of a new key and decryption with that key to go forward. At such time as a user agent 118 for the host 104 requests playback, the host 104 goes back to the content server 102 to download key segments, regenerate the key and then decrypt and play back the encrypted cached streamed content (at 220-226).

**[0026]** If, however, after requesting playback the content server is unavailable or does not wish to allow playback, then the key segments that have been requested for download will not be downloaded thus preventing the host from generating a new key and decrypting or playing the encrypted streamed content (at 220, 228). As noted above, reasons for the content server 102 to refuse further download of the streamed content could be implementation of a takedown requirement because a copyright owner had identified infringement and notified the entity controlling the content server 102 or could be based on the fact that the content server 102 identifies that the host 104 is no longer in a geographic region

for which playback is permitted for that particular content. Information as to the geographic region of the host would generally be determined by the content server 102 based on the location of the server through which the host 104 sent the request for playback time. Although it is contemplated that the host 104 may send its own location information in other implementations, the location of the server through which the host makes the playback request is considered more reliable and less prone to manipulation. The Internet protocol (IP) address of the server used by the host 104 to query the content server 102 may be used to locate the server and provide the necessary geographic location to determination at the content server whether that geographic location is permitted to receive the requested streamed content. Thus, the cached streamed content stored in the storage device is protected by an encryption key, which may be generated from the streamed content itself as discussed below, so that the content server controls the ability of the host 104 to replay the cached streamed content by controlling access to the streamed content used to generate the key when playback is subsequently requested by the user agent 118.

**[0027]** Turning to FIG. 3, a method of using multiple segments of streamed data to create an encryption key for the streamed data is shown. The method of FIG. 3 may be executed by the virtual storage device of the client in the host. Upon a request by a user agent 118 either directly or as a proxy for a content server 102 to create an encryption key k (at 302), the process of creating the key k is initiated where streamed data with which to create key k is split into n portions of which t are required to reconstruct k, where $t < n$ (at 304). This may be done using a secret sharing algorithm such as Shamir (as described in "How to share a secret", *Communications of the ACM* 22 (11): 612–613), or Asmuth-Bloom ("A modular approach to key safeguarding", C.A. Asmuth and J. Bloom. *IEEE Transactions on Information Theory*, IT-29(2):208-210, 1983), among others. First, n key segments $s_{1...n}$ are retrieved from the content server (at 306), where each segment is retrieved using a reliable or unreliable protocol. As noted above, a reliable protocol may be a TCP-based protocol where, if a packet is dropped, it is

re-requested and corrected. In contrast, an unreliable protocol may be the UDP-type protocol where there is no guarantee of reliability. After retrieving the n key segments, the host may determine if the protocol is a reliable protocol (at 308). If a reliable protocol is detected then each portion of encryption key k is encrypted with the corresponding segment s and the encrypted key k(s) is stored with the encrypted streamed content in the storage device (at 312, 314).

**[0028]** If instead of a reliable protocol, an unreliable protocol is detected such as UDP, then each segment $s_{1...n}$ is retrieved from the content server again, to verify each segment. Segments which are not verified are retrieved (e.g. which do not repeatably match) again until verified (at 310). After verification, the next steps of encrypting each portion of key k with corresponding segment s and then storing encrypted key k(s) persistently with encrypted content is undertaken (at 312, 314).

**[0029]** Once the unencrypted key is created, the key k is only maintained in the host 104 long enough to encrypt the streamed content and then is discarded rather than saved in the storage device 108 or host 104. Referring again to FIG. 2, at the point where the user agent 118 initiates playback by requesting key segments to again be downloaded, assuming that there has been no takedown provision enforced by the content server or other restriction that prevents download, the key can then be again derived using an unreliable protocol such as UDP as shown in FIG. 4. The first step in the process is to retrieve the encrypted key k(s) and then retrieve n key segments $s_{1...n}$ from the content server where each segment is retrieved using an unreliable protocol (at 402, 404). Each portion of the key k(s) is decrypted with each respective segment $s_{1...n}$ (at 406). The unencrypted key k is then reconstituted from the elements $k_{1...n}$ that are recovered from this step using a secret sharing algorithm as noted in FIG. 3 at step 304, where t portions are required to reconstruct k and where $t<n$ (at 407). After decrypting the key k(s) the streamed content is decrypted with k and played back by the user agent 118 on the host 104 (at 408). In one implementation it is preferred that, for the encryption and decryption process of FIGS. 3-4 to work properly, that a threshold value t (i.e.

the minimum number of portions required to reconstruct unencrypted encryption key k) should be set sufficiently low to allow for errors in the unreliable stream. In general, parameters t and n are known parameters to the client 112 of the user agent 118 and are a constant and also known to the host 104.

**[0030]** A secret sharing protocol may be used for the encryption and decryption mechanism to allow for both reliable and unreliable download protocols. Although any of a number of encryption protocols known to those of skill in the art may be implemented, one suitable secret sharing protocol is the Shamir secret sharing scheme. The key generation process of FIG. 3, as well as the key re-generation process described in FIG. 4, involve splitting the key k into multiple segments ($k_{1...n}$) using a secret sharing technique so that the encrypted key k(s) may be processed one segment at a time with the corresponding streamed data segment $s_{1...n}$ obtained from the content server. Also, the key utilized in the embodiments of FIGS. 3-4 and generated with a secret sharing protocol is dependent on the bit rate and so the key segments are intentionally downloaded by the host at a bit rate that is lower than the highest available bit rate that can be rendered on the host 104 under the assumption that a lower bit rate may only be available when a user decides to play back previously cached content. This assumed bit rate available for later download will always be lower to allow for more flexibility if the later bit rate is indeed a low bit rate. In an alternative embodiment, the content to be cached may be intentionally downloaded by the host at a bit rate that is lower than the highest available bit rate that can be rendered by the host, while the key utilized in the embodiments of FIGS. 3-4 may be created from a portion of the same content downloaded at a higher bit rate than the bit rate of the content to be cached. Using a different bit rate for the key than for the content reduces the predictability of the key and may therefore increase the strength of the encryption.

**[0031]** Again, in a situation where a reliable protocol is available, the ordinary mechanism for key generation may be to hash the content to create a key and, because it's a reliable protocol, the same key will be guaranteed each time because

the data downloaded is reliable and so no splitting up of the key into portions and segments is necessary. Because and unreliable protocol such as UDP is by definition unreliable, the bits downloaded at any given time may not match all the bits downloaded previously such that only a certain number of segments of the key may be identical to previously downloaded segments of the key. By using multiple segments as described above and relying on repeated download of the same bits and comparison to the previously downloaded bits until enough matches are made, a high confidence, although not a guarantee, that an accurate key has been generated may be developed for instances where only an unreliable protocol is available.

**[0032]** Although the description above refers to actions executed by the processor of the host following instructions of the client for user agent, embodiments are also contemplated in which the storage device (built-in, removable from, or external to, the host) may include built-in cryptographic capabilities, secure storage for keys and a pseudo-random or true random number generator that can be used to generate encryption keys. For example, referring to FIG. 5, an alternative system is shown where a content server 502 in a network 506 may provide a low bit rate stream 524 or a high bit rate stream 526 via a web server 528 to the host 504 via a wired or wireless communication link 503. The host 504 includes a playback application that acts as a user agent 518, a storage device 508 and a client application 512 in communication with a host processor 510. Similar to the embodiment of FIG. 1 discussed above, the user agent 518 may include one or more virtual storage devices (VSD) 514 and a caching engine 516. The caching engine may include a queue manager 520 and a policy database 522 for handling downloads. The storage device 508 of FIG. 5 includes cache storage 530, a secure key storage 532, and a processor 534. As described below, although the storage device 508 may be recited as carrying out particular steps, in one embodiment it is the processor 534 of the storage device 508 carrying out the recited steps in conjunction with firmware or software resident in the processor 534 or the storage device 508 generally. Also, although the host 504 is shown

with an internal storage device, as mentioned above, in various embodiments the storage device 508 can be embedded, removable or external to the host 504.

**[0033]** In this embodiment, while it is assumed that the storage device 508 cannot issue a direct HTTP request, the storage device 508 can operate a user agent 518 to do so on its behalf. In this embodiment, the processor 534 of the storage device 508 may be configured to provide a secure location for the generation of keys. The processor may also be configured to transparently decrypt content when a key is provided and provide this decrypted content to the host 504 for playback. Furthermore, the storage device 508 of the embodiment of FIG. 5 may derive the key from a provided stream in a manner somewhat similar to that executed by the processor 110 of the host 104 in the embodiment of FIG. 1. A key stream is a low bit rate stream requested for the purpose of deriving key segments as describe previously. The actual algorithm used to select key segments from the key stream may be unique to the storage device 508 and does not need to be known to the host 504. For example, the algorithm may be instructions to take certain parts of the streamed content, such as every third 4k segment from a 64k stream or based on some other pattern, based on a hash algorithm or any of a number of other parameters. Because neither the storage device processor 534 nor the content server 502 can initiate commands, the user agent 518 may drive communication with both the server and the storage device 504. The user agent 518 may initiate the process as shown.

**[0034]** Referring to FIG. 6, the storage device 508 of may work with the user agent 518 to create a key by first requesting the allocation of a new encryption key (at 602) and then beginning the process of creating the new encryption key (at 604). In the same manner described in the host version of FIG. 3, although now performed by the storage device 508 rather than the host 104, the key k is split into n portions, t of which are needed to reconstruct k, where $t<n$ (at 606). The user agent 518 then requests the required key stream size from the storage device 508, the required stream size dependent on the needs of the particular key segment selection algorithm utilized by the host where the required key stream size may be

communicated to the user agent 118 by the virtual storage device 114 (at 608). As in the embodiment of FIG. 3, the client application 512 on the host 504 determines if the protocol between host 504 and content server 502 is a reliable or unreliable format (at 610). If reliable, then the user agent 518 will receive the stream (having the stream size identified by the storage device 508 as necessary) only one time and will send that stream on to the storage device 508 (at 612, 616). If the protocol is determined by the user agent 518 to be unreliable, the stream packets making up the key stream size will be retrieved multiple times until verified as described previously and then sent to the storage device 508 once the packets of the stream are consistently received (at 614, 616).

**[0035]** Once the key stream has been received at the storage device 508, the storage device 508 will then select key segments $s_{1...n}$ from the key stream, encrypt each portion of key k with the segments s, store encrypted key k(s) in the secure key storage area 532 and use k to encrypt the streamed data (at 618-622). Playback may be accomplished using the same mechanism as the host-only environment described with respect to FIG. 4, with the exception that the key stream is provided to the storage device 508 as-is and the storage device 508 does the key derivation and manipulation.

**[0036]** In addition to the host-only processing and storage device processing embodiments for using reliable or unreliable protocols to generate an encryption key using one or more segments of the streamed content itself, additional embodiments are contemplated where the key generated from the streamed content is combined with another key from another source, where this key from another source is hereinafter referred to as an overlay key. For example, in one embodiment the overlay key may be a hardware key derived from a subscriber identity module (SIM) card or host hardware to also prevent the streamed content cached in the storage device from being copied to another storage device. Using a hardware key derived from the SIM card would provide the network operator that issues the SIM card the ability to control a subscriber based on subscriber status. This added protection for the network operator prevents a user from switching

memory cards and unlocking the streamed content cached in the memory card. Alternatively, the overlay key may be a host binding key so that rather than binding the cached streamed data to a SIM card, the key derived from the streamed content as previously described would be combined with a key created by host-unique parameters. Examples of such parameters include host serial number, chipset information for the host, international mobile equipment identity (IMEI) information and so on. Furthermore, it is contemplated that both a host-specific key and a SIM-based key may be combined with the key derived from the streamed content in another embodiment.

**[0037]** An example of how the overlay key (or keys) may be integrated into the key generation process of FIG. 3 is illustrated in FIG. 7. The method of FIG. 7 may be executed by the virtual storage device 114 of the client 112 in the host 104 or by the storage device 508 in conjunction with the user agent 518. Upon a request by the user agent to start the process to create an encryption key k (at 702) the initial streamed data with which to create key k is split into end portions of which t are required to reconstruct k, where $t < n$ (at 704). The n key segments $s_{1...n}$ are retrieved from the content server (at 706), where each segment is retrieved using a reliable or unreliable protocol. At this point, the host will overlay each segment s with the overlay key o (at 708). The overlay may be accomplished applying an exclusive or (XOR) function to the overlay key o and each segments s, by applying any of a number of known hash functions to the overlay key and each segment s, or through other known methods. After retrieving the n key segments, the host may determine if the protocol is a reliable protocol (at 710). If a reliable protocol is detected then each portion of encryption key k is encrypted with s(o), the corresponding segment overlayed with the overlay key o, and the encrypted key k(s(o)) is stored with the encrypted streamed content in non-volatile memory the storage device (at 714, 716). As noted above with respect to FIGS. 3-4, the key k is actually broken up into multiple portions $k_{1...n}$ so that each portion of the split key is combined with a respective portion of $s(o)_{1...n}$ to obtain k(s(o))

$_{1...n}$. In other embodiments, overlay key o may also be broken up using a secret sharing algorithm where t!=n, but this is not necessary, as t is always equal to n.

**[0038]** If instead of a reliable protocol, an unreliable protocol is detected such as UDP, then each segment $s_{1...n}$ is retrieved from the content server again, to verify each segment and then overlayed with overlay key o. Segments which are not verified are retrieved again until verified (at 712). After verification, the next steps of encrypting each portion of key k with corresponding overlayed segment s(o) and then storing k(s(o)) persistently in the storage device with encrypted streamed content are undertaken (at 714, 716).

**[0039]** Similar to the method for regenerating the key discussed in FIG. 4 above, regeneration of the key from encrypted key k(s(o)) is accomplished in the process shown in FIG. 8. First, the host processor retrieves the encrypted key from the storage device. In this case, the encrypted key created by the overlay process in FIG. 7 is k(s(o)). Thus, in response to a request to play back the streamed data that has been cached in the storage device associated with the host, the first step for regenerating the key is to retrieve the encrypted key k(s(o)) (at 802). Next, the n key segments $s'_{1...n}$, where $s'_{1...n}$ represent what could be segments with slight variation in the bits contained therein as compared to the original $s_{1...n}$ due to use of an unreliable protocol, are retrieved from the content server, where each key segment is retrieved using an unreliable protocol (at 804). Each segment $s'_{1...n}$ is overlaid with the overlay key o (at 806) obtained from the SIM card or host hardware. The overlay process for each segment with key o may be accomplished by use of an XOR function, a hash or other technique. After overlaying each segment $s'_{1...n}$ to obtain $s'(o)_{1...n}$, each portion of $k_{1...n}(s(o))$ is decrypted with $s'(o)_{1...n}$ (at 808). The result of this decryption is to obtain unencrypted key k which is then used to decrypt the streamed content in the storage device (at 810).

**[0040]** Alternative methods for implementing the overlay of an overlay key such as a SIM key or host-specific key, is to introduce the overlay key at a different point in the encryption process. Accordingly, the encryption mechanism

using an overlay key illustrated in FIG. 7 may be modified to remove the overlay step of step 708 and add in the overlay later in the process. For example, in this alternative overlay embodiment, the overlay which was previously accomplished in step 708 may be accomplished by encrypting each portion of k with s in step 714 rather than encrypting each portion of k with s(o). By encrypting each portion of k with s rather than s(o) the result would be an encrypted key k(s) which may be stored persistently with encrypted content and the overlay of o with k may be done separately in parallel with encryption of k with s such that k(s) and k(o) are maintained separately. The encryption of the received data then would be done by k(o) which would not be saved or stored. Accordingly, in the alternative overlay encryption mechanism to what is currently shown in claim 7, k(s) is stored but the streamed data is stored and encrypted by k(o).

**[0041]** The regeneration of this alternative version of overlay encryption with an overlay key o is shown in FIG. 9 and differs somewhat from that shown in FIG. 8. The regeneration of key k would first include retrieving the encrypted key k(s) (at step 802) from the storage device and then retrieving the n key segments $s'_{1\ldots n}$ from the content server, where each segment is retrieved using an unreliable protocol, and then overlaying $s'_{1\ldots n}$ with overlay key o to create $s'(o)_{1\ldots n}$ (at step 804). Each portion of $k(s)_{1\ldots n}$ may then be decrypted with $s'(o)_{1\ldots n}$ (at step 906). This would lead to obtaining the unencrypted key k which may then be combined with the overlay key o to produce k(o) (at step 908). Finally, the content encrypted with k(o) in the alternative overlay encryption process discussed above may be decrypted with the now-obtained key k(o) (at 910).

**[0042]** The techniques and devices described above, whether for non-overlay encryption or overlay encryption, may be configured such that the entirety of the encryption and decryption process is handled by the host or by the storage device. In alternative embodiments, responsibility for portions of the process of encryption and decryption of the streamed content may be shared between the host and storage device, however additional care would be necessary regarding

handoffs of intermediate encryption and decryption values between the host and storage device in such alternative embodiments.

**[0043]** Any of a variety of mobile device platforms may be used to implement the streamed data protection methods and systems disclosed above. For example a video playback device utilizing a Google Android-based platform is contemplated. In such an implementation, the mobile device may store downloaded content from a video streaming site where streamed content is cached in local storage on a removable storage card and is encrypted using a key derived from streamed content retrieved. As with the embodiments described above, the mobile device (host) would not retain a full, unencrypted key, such that playback is impossible if the content server is not available or is enforcing an access restriction policy.

**[0044]** Methods and systems have been disclosed for permitting a content server to have more control over streamed content that is cached locally to a storage device associated with a host, such as a portable playback device, rather than immediately consumed at the host. The method may include creating an encryption key from a portion of the streamed content itself and encrypting the streamed content with that key as the content is stored to the storage device associated with the host. The method also includes procedures for handling streamed data over unreliable protocols or reliable protocols. Additionally, an overlay key relating to the SIM card and/or host device may be combined with the encryption key derived from a portion of the streamed content to permit further control over content distribution in addition to the control provided to the content server to takedown or enforce access limitations to streamed content. The methods may be implemented on a host device and carried out by a processor on the host device executing applications residing on the host, or on both a storage device configured to manage key generation and encryption in coordination with the host device it is connected to.

WHAT IS CLAIMED IS:

1. A method for protecting streamed content stored locally on a storage device associated with a host, the method comprising:

in a processor of the host:

requesting streamed content from a content server;

downloading a portion of the streamed content at a predefined bit rate from the content server;

creating an encryption key from the portion of the streamed content downloaded at the predefined bit rate;

downloading the streamed content from the content server at a second bit rate different from the predefined bit rate;

encrypting the streamed content downloaded at the second bit rate with the encryption key created from the portion of the streamed content downloaded at the predefined bit rate; and

storing the encrypted streamed content on the storage device, wherein the encrypted streamed content is cached for playback from the storage device and protected from unauthorized playback.

2. The method of claim 1, wherein the predefined bit rate is lower than the second bit rate.

3. The method of claim 1, wherein the predefined bit rate is higher than the second bit rate.

4. The method of claim 1, wherein creating the encryption key includes determining whether the portion of content downloaded at the predefined rate has been downloaded using an unreliable protocol and, responsive to determining that the portion of content has been downloaded using an unreliable protocol, verifying an accuracy of the portion of content.

5. The method of claim 1, wherein creating the encryption key further comprises:

dividing the encryption key into a plurality of key portions;

retrieving a corresponding plurality of key segments at the pre-defined bit rate;

determining whether the corresponding plurality of retrieved key segments were retrieved using a reliable data protocol or the unreliable data protocol; and

when the processor determines that the plurality of retrieved key segments were sent with an unreliable data protocol, retrieving the corresponding plurality of key segments from the content server again to verify an accuracy of the plurality of key segments.

6. The method of claim 5, further comprising encrypting the encryption key with the corresponding plurality of key segments for storage with the streamed content on the storage device.

7. The method of claim 6, further comprising discarding an unencrypted version of the encryption key after encrypting the streamed content with the encryption key.

8. The method of claim 7, further comprising:

receiving a request to playback the streamed content stored in the storage device and encrypted with the encryption key;

retrieving the encrypted encryption key from the storage device;

retrieving the plurality key segments from the content server;

decrypting the encrypted encryption key with the plurality of key segments retrieved from the content server;

retrieving from the storage device the streamed content encrypted with the encryption key;

and

decrypting the encrypted streamed content stored using the decrypted encryption key, wherein playback of the streamed content stored on the storage device is dependent upon accessing the portion of the streamed content again from the content server to recreate the encryption key that was used to encrypt the streamed content when it was previously stored.

9. The method of claim 5, further comprising:

applying an overlay key to the plurality of retrieved key segments to create an overlaid plurality of key segments; and

encrypting the encryption key with the overlaid plurality of retrieved key segments, wherein decryption of the encryption key requires access to both the overlay key and the plurality of retrieved key segments.

10. The method of claim 9, wherein applying the overlay key comprises applying an overlay key provided by a network operator.

11. The method of claim 9, wherein applying the overlay key comprises applying a host-specific key to the plurality of retrieved key segments.

12. A host device for protecting streamed content stored locally on a storage device in communication with the host, the host device comprising:

an interface for communicating with a remotely located content server; and

a processor in communication with the interface, the processor configured to:

request streamed content from the content server;

create an encryption key for the streamed content from a portion of the streamed content downloaded from the content server at a predefined bit rate;

download the streamed content at a second bit rate different from the pre-defined bit rate;

encrypt the content downloaded at the second bit rate with the encryption key created from the portion of the streamed content downloaded at the predefined bit rate for storage on the storage device; and

store the encrypted streamed content on the storage device, wherein the encrypted streamed content is cached for playback from the storage device and protected from unauthorized playback.

13. The host device of claim 12, wherein the predefined bit rate is less than the second bit rate.

14. The host device of claim 13, wherein the predefined bit rate is greater than the second bit rate.

15. The host device of claim 12, wherein to create the encryption key the processor is further configured to:

divide the encryption key into a plurality of portions;

retrieve a corresponding plurality of key segments at the predefined bit rate;

determine whether the corresponding plurality of retrieved key segments were sent with a reliable data protocol or an unreliable data protocol; and

when the processor determines that the corresponding plurality of retrieved key segments were sent with an unreliable data protocol, retrieve the corresponding plurality of key segments from the content server again to verify an accuracy of the plurality of key segments.

16. The host device of claim 15, wherein the processor is further configured to encrypt the encryption key with the corresponding plurality of key segments for storage with the content on the storage device.

17. The host device of claim 16, further comprising discarding an unencrypted version of the encryption key after encrypting the streamed content with the encryption key.

18. The host device of claim 17, wherein the processor is further configured to:

receive a request to playback the streamed content stored in the storage device and encrypted with the encryption key;

retrieve the encrypted encryption key from the storage device;

retrieve the plurality of key segments from the content server using an unreliable protocol;

decrypt the encrypted encryption key with the plurality of key segments retrieved from the content server;

retrieve from the storage device the streamed content encrypted with the encryption key;

and

decrypt the encrypted streamed content retrieved from the storage device using the decrypted encryption key, wherein playback of the streamed content stored on the storage device is dependent upon accessing the portion of the streamed content again from the content server to recreate the encryption key that was used to encrypt the streamed content when it was previously stored.

19. The host device of claim 15, wherein the processor is further configured to:

apply an overlay key to the plurality of retrieved key segments to create an overlaid plurality of key segments; and

encrypting the encryption key with the overlaid plurality of key segments, wherein decryption of the encryption key requires access to both the overlay key and the plurality of retrieved key segments.

20. The host device of claim 19, wherein the overlay key comprises an overlay key provided by a network operator.

21. The host device of claim 19, wherein the overlay key comprises applying a host-specific key to the plurality of retrieved key segments.

22. The host device of claim 21, wherein the host-specific key is based on at least one of a host serial number, chipset information for the host and IMEI information.

23. A method for protecting streamed content stored locally on a storage device associated with a host, the method comprising:

in a processor of the host:

receiving a request to playback streamed content stored on the storage device, wherein the streamed content stored on the storage device was previously downloaded at a predefined bit rate and has been encrypted with an encryption key comprising a portion of the streamed content previously downloaded at a second bit rate different than the predefined bit rate;

retrieving an encrypted version of the encryption key from the storage device;

retrieving a plurality of key segments from a content server at the second bit rate;

decrypting the encrypted encryption key with the plurality of key segments retrieved from the content server;

reconstituting the encryption key with a secret sharing algorithm;

retrieving from the storage device the streamed content encrypted with the encryption key; and

decrypting the streamed content stored on the storage device using the decrypted and reconstituted encryption key, wherein playback of the streamed content stored on the storage device is dependent upon accessing the portion of the streamed content again from the content server to recreate the encryption key that was used to encrypt the streamed content when it was previously stored.

24. The method of claim 23, wherein the predefined bit rate is less than the second bit rate.

25. The method of claim 23, wherein the predefined bit rate is greater than the second bit rate.

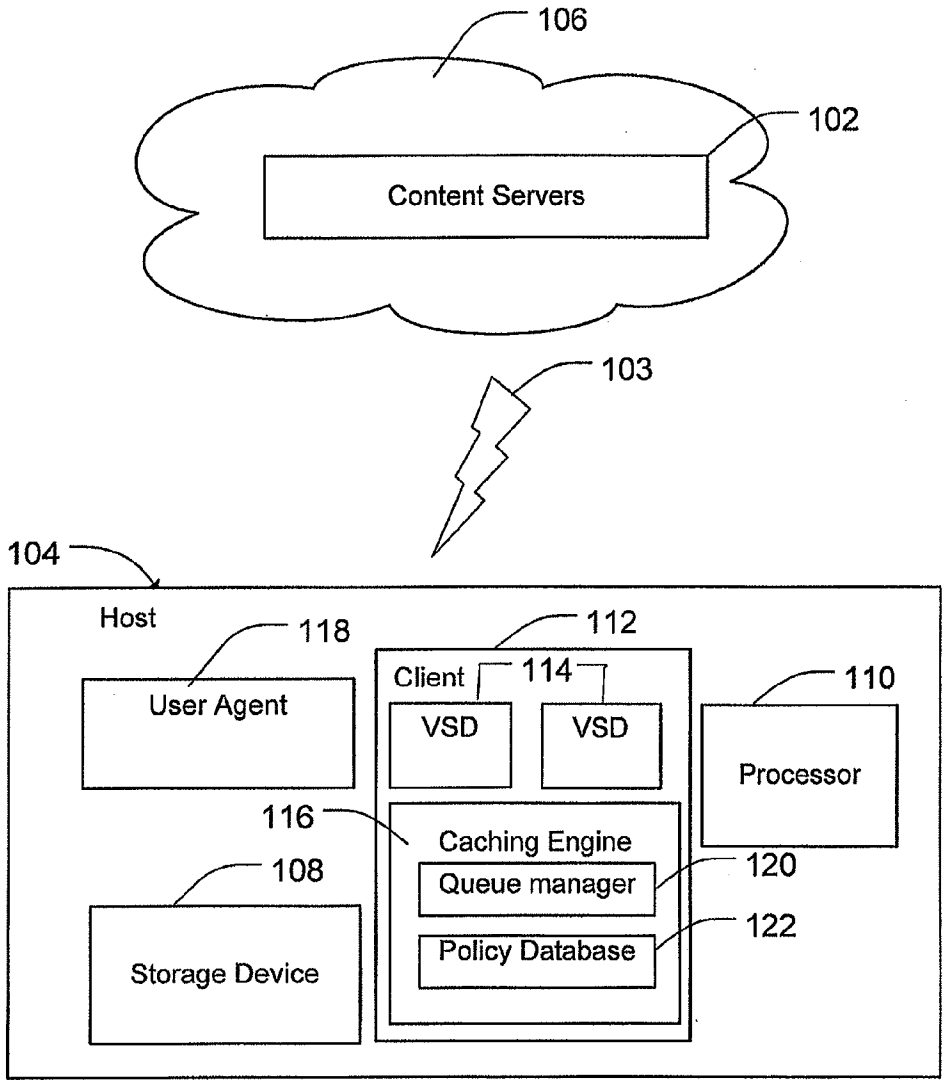
106
102
Content Servers
103
104
Host
118
User Agent
112
Client
114
VSD
VSD
110
Processor
116
Caching Engine
Queue manager
120
122
Policy Database
108
Storage Device

FIG. 1

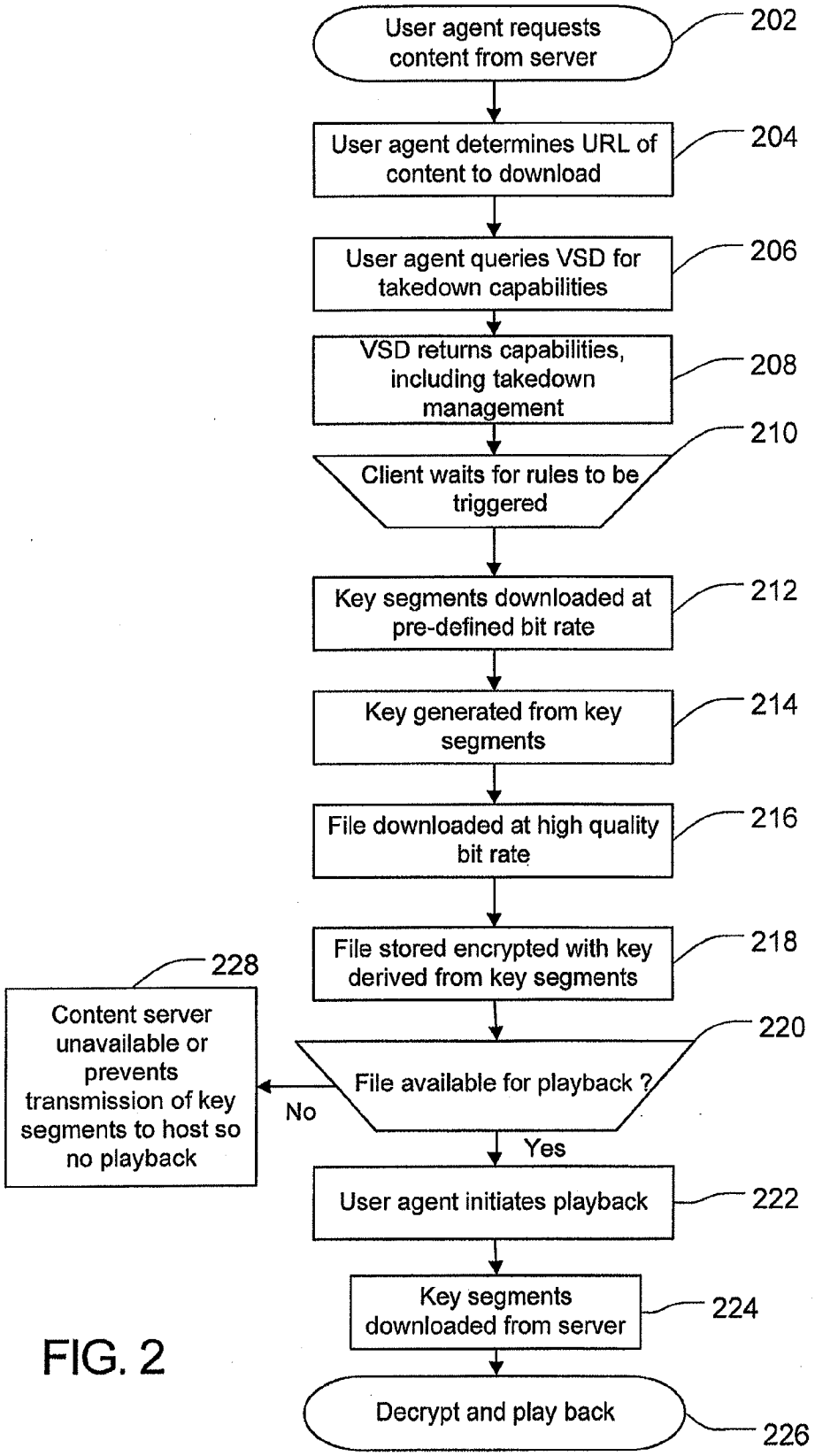
User agent requests content from server
202
User agent determines URL of content to download
204
User agent queries VSD for takedown capabilities
206
VSD returns capabilities, including takedown management
208
Client waits for rules to be triggered
210
Key segments downloaded at pre-defined bit rate
212
Key generated from key segments
214
File downloaded at high quality bit rate
216
File stored encrypted with key derived from key segments
218
File available for playback ?
220
No
Content server unavailable or prevents transmission of key segments to host so no playback
228
Yes
User agent initiates playback
222
Key segments downloaded from server
224
Decrypt and play back
226

FIG. 2

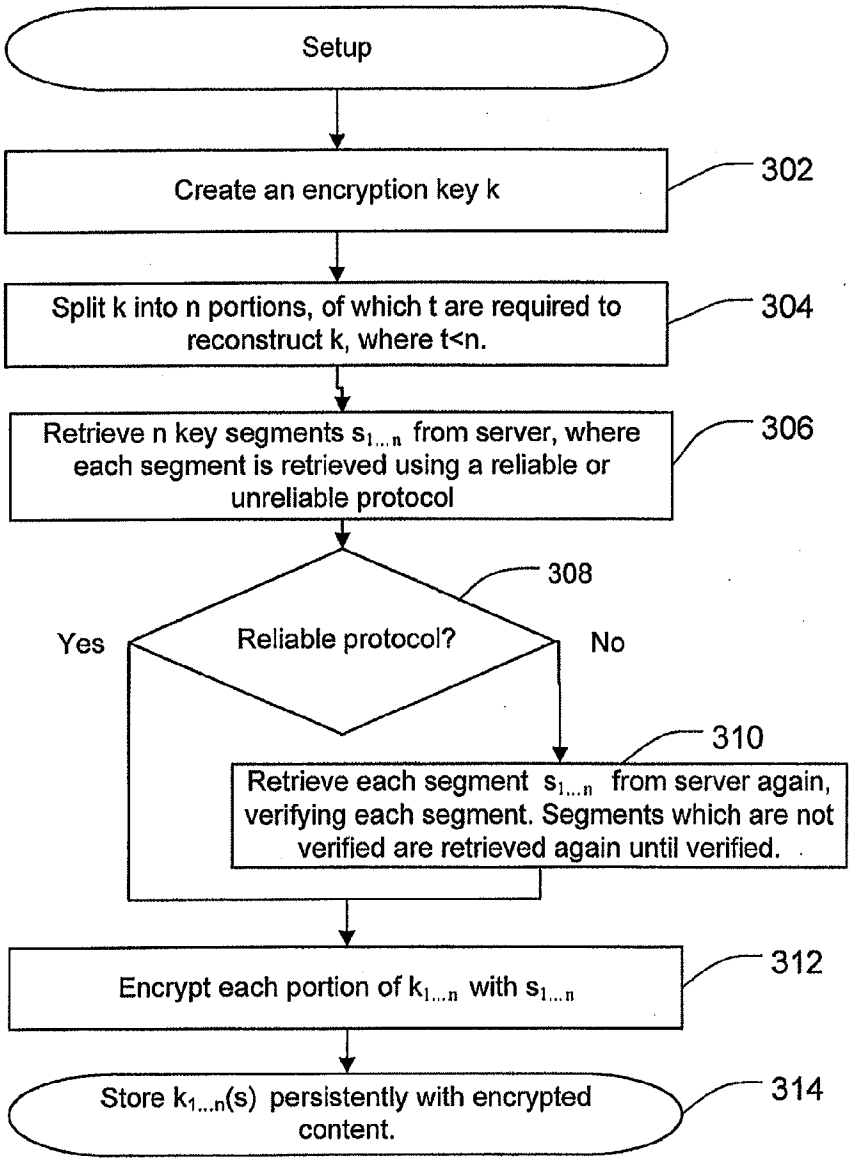
Setup
Create an encryption key k
302
Split k into n portions, of which t are required to reconstruct k, where t<n.
304
Retrieve n key segments $s_{1...n}$ from server, where each segment is retrieved using a reliable or unreliable protocol
306
308
Reliable protocol?
Yes
No
310
Retrieve each segment $s_{1...n}$ from server again, verifying each segment. Segments which are not verified are retrieved again until verified.
Encrypt each portion of $k_{1...n}$ with $s_{1...n}$
312
Store $k_{1...n}$(s) persistently with encrypted content.
314

FIG. 3

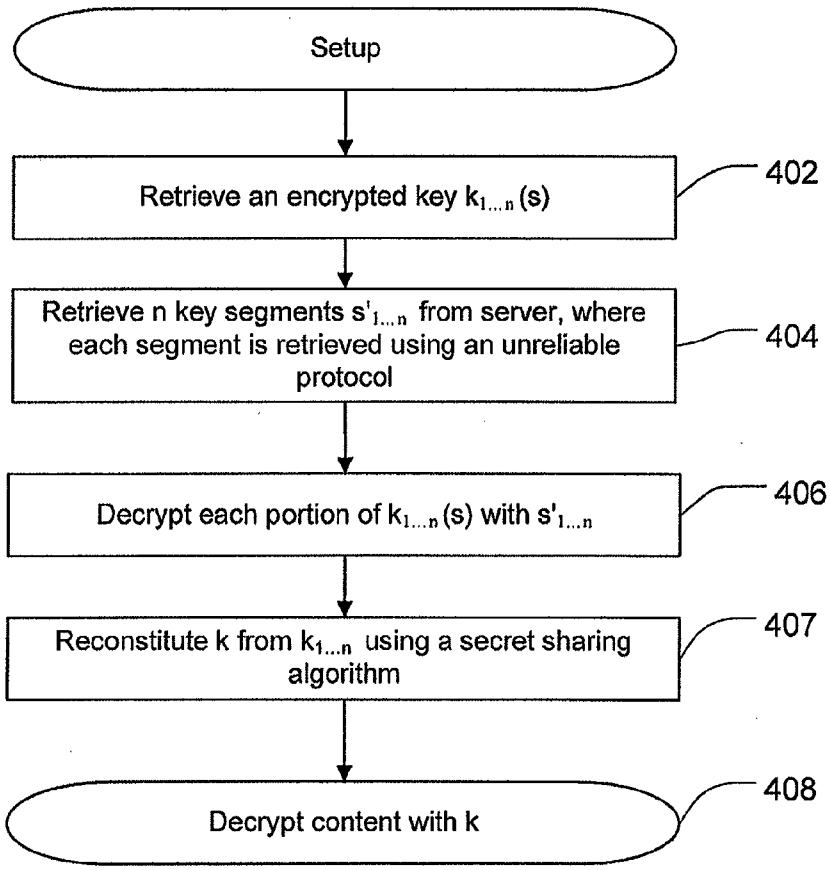
Setup
Retrieve an encrypted key $k_{1...n}$ (s)
402
Retrieve n key segments $s'_{1...n}$ from server, where each segment is retrieved using an unreliable protocol
404
Decrypt each portion of $k_{1...n}$ (s) with $s'_{1...n}$
406
Reconstitute k from $k_{1...n}$ using a secret sharing algorithm
407
Decrypt content with k
408

FIG. 4

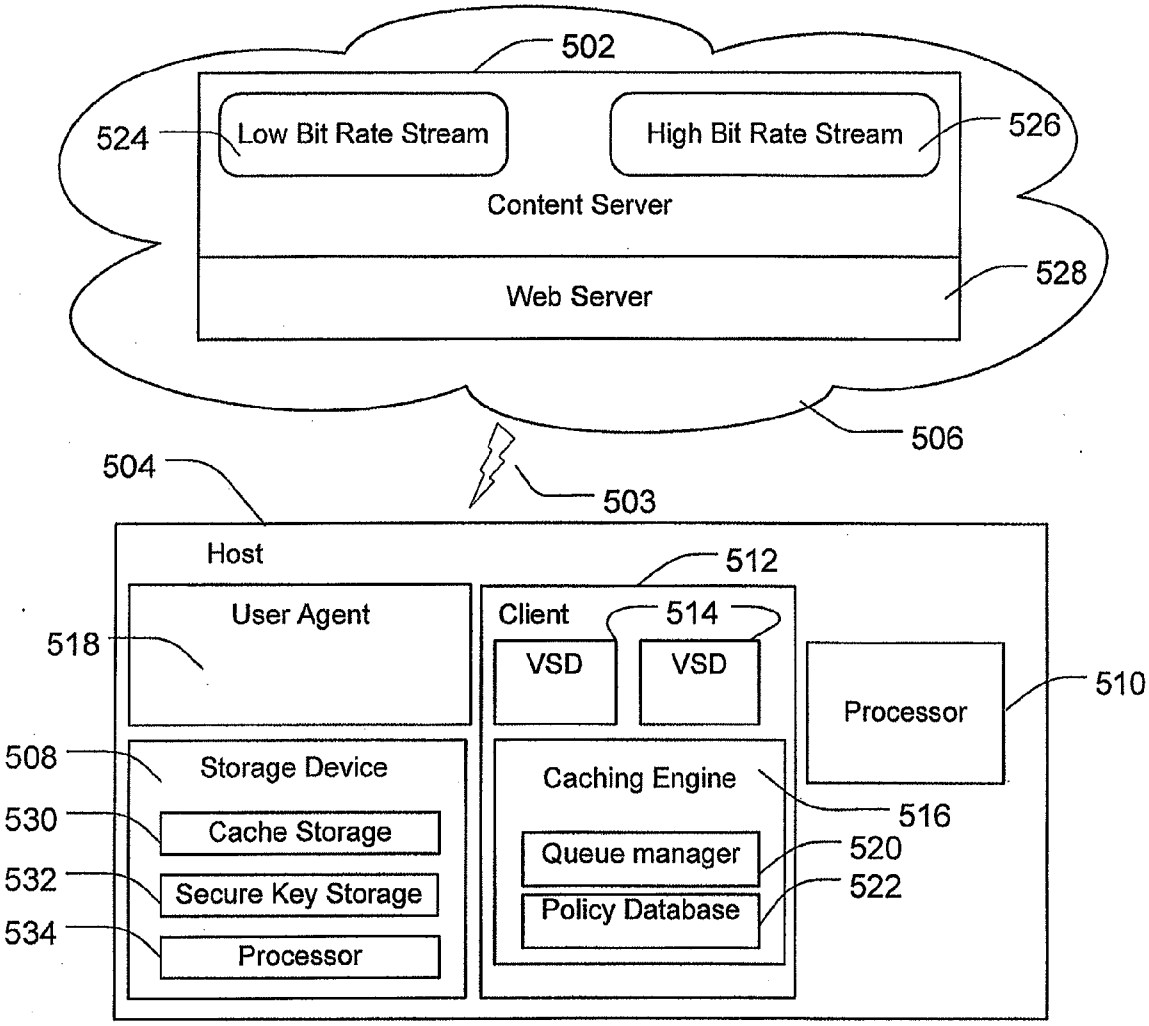
502
524
Low Bit Rate Stream
High Bit Rate Stream
526
Content Server
Web Server
528
506
504
503
Host
512
User Agent
518
Client
514
VSD
VSD
Processor
510
508
Storage Device
530
Cache Storage
532
Secure Key Storage
534
Processor
Caching Engine
516
Queue manager
520
Policy Database
522

FIG. 5

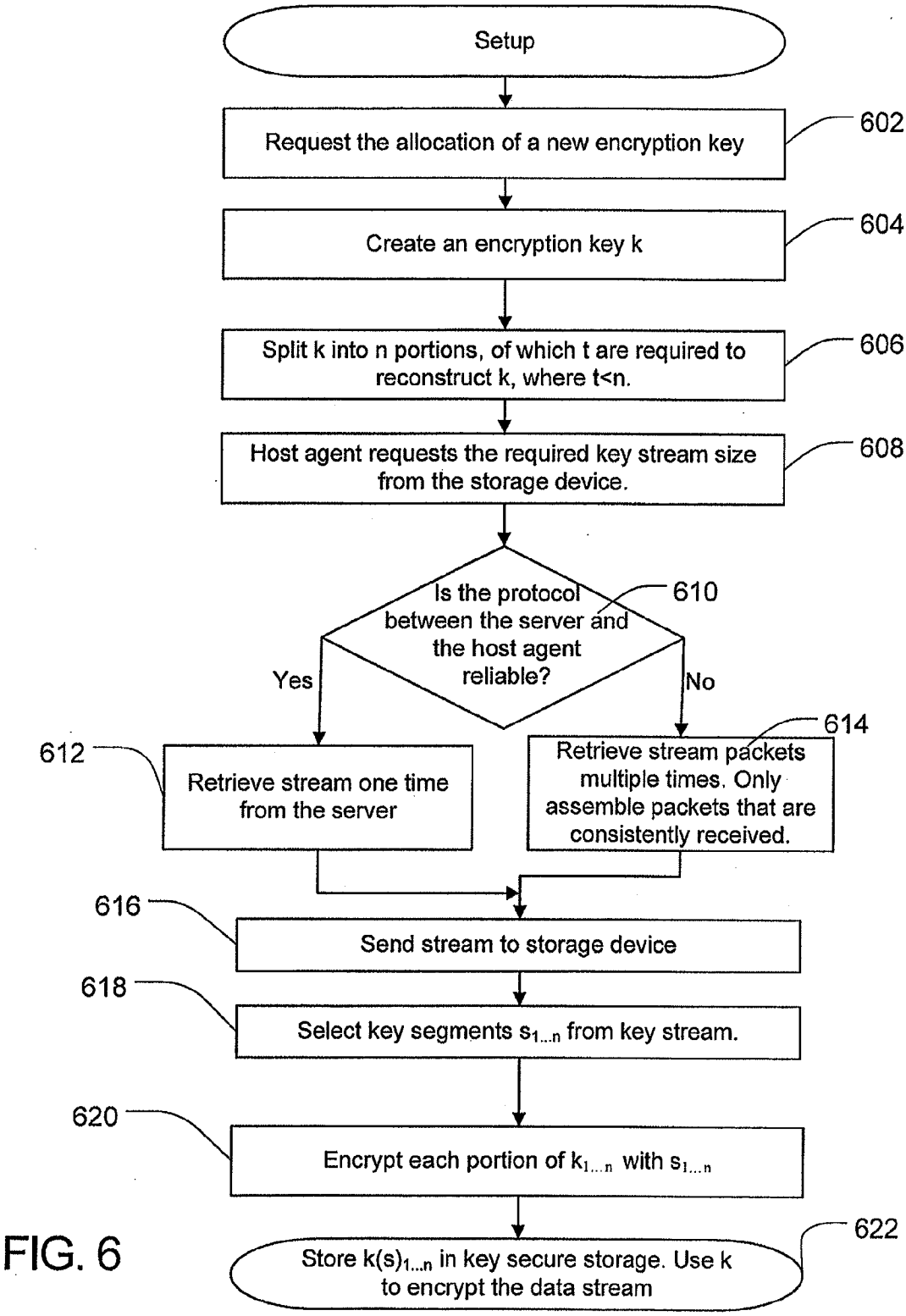
Setup
Request the allocation of a new encryption key
602
Create an encryption key k
604
Split k into n portions, of which t are required to reconstruct k, where t<n.
606
Host agent requests the required key stream size from the storage device.
608
Is the protocol between the server and the host agent reliable?
610
Yes
No
612
Retrieve stream one time from the server
614
Retrieve stream packets multiple times. Only assemble packets that are consistently received.
616
Send stream to storage device
618
Select key segments $s_{1...n}$ from key stream.
620
Encrypt each portion of $k_{1...n}$ with $s_{1...n}$
Store $k(s)_{1...n}$ in key secure storage. Use k to encrypt the data stream
622

FIG. 6

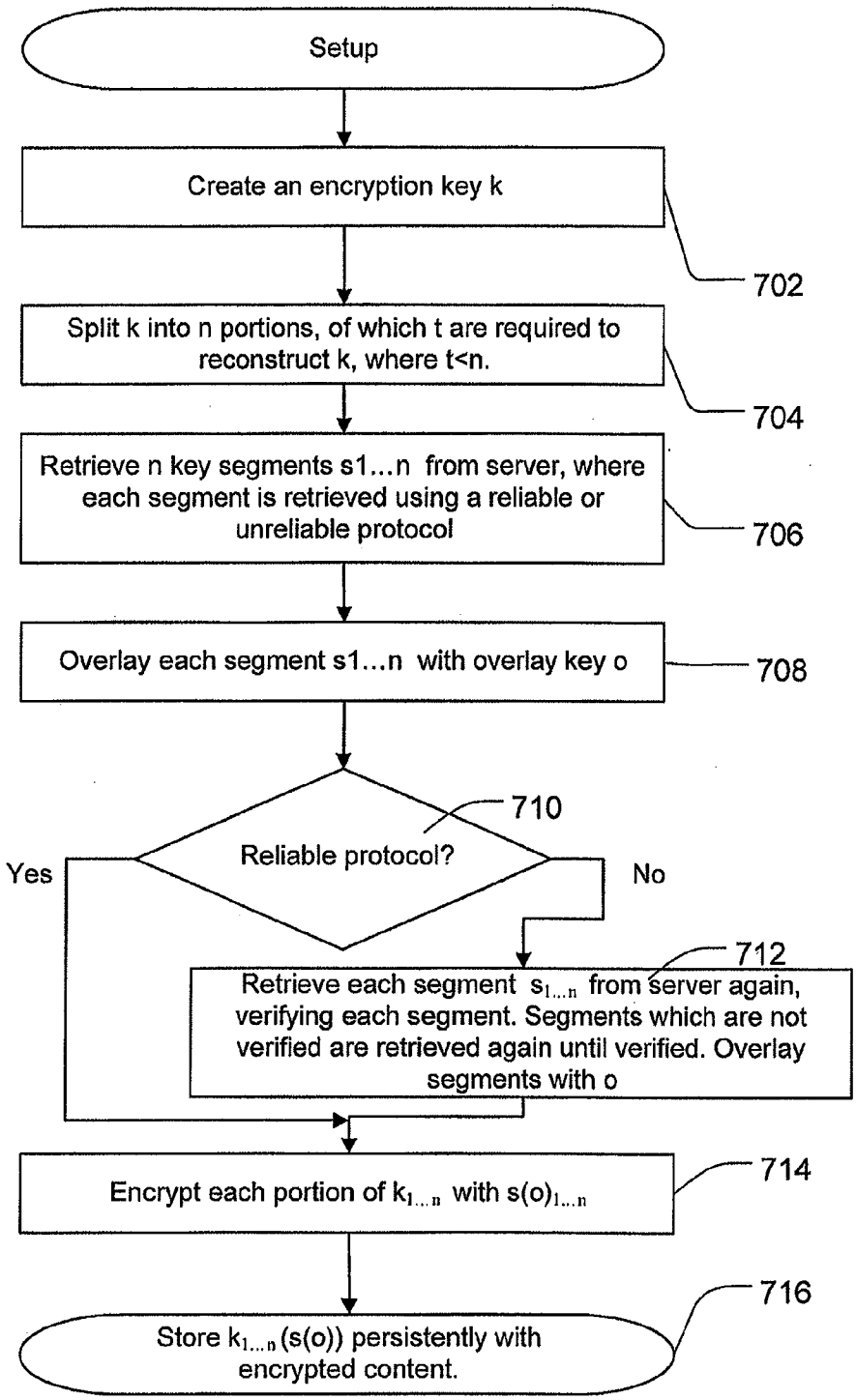
Setup
Create an encryption key k
702
Split k into n portions, of which t are required to reconstruct k, where t<n.
704
Retrieve n key segments s1...n from server, where each segment is retrieved using a reliable or unreliable protocol
706
Overlay each segment s1...n with overlay key o
708
710
Reliable protocol?
Yes
No
712
Retrieve each segment $s_{1...n}$ from server again, verifying each segment. Segments which are not verified are retrieved again until verified. Overlay segments with o
Encrypt each portion of $k_{1...n}$ with $s(o)_{1...n}$
714
716
Store $k_{1...n}(s(o))$ persistently with encrypted content.

FIG. 7

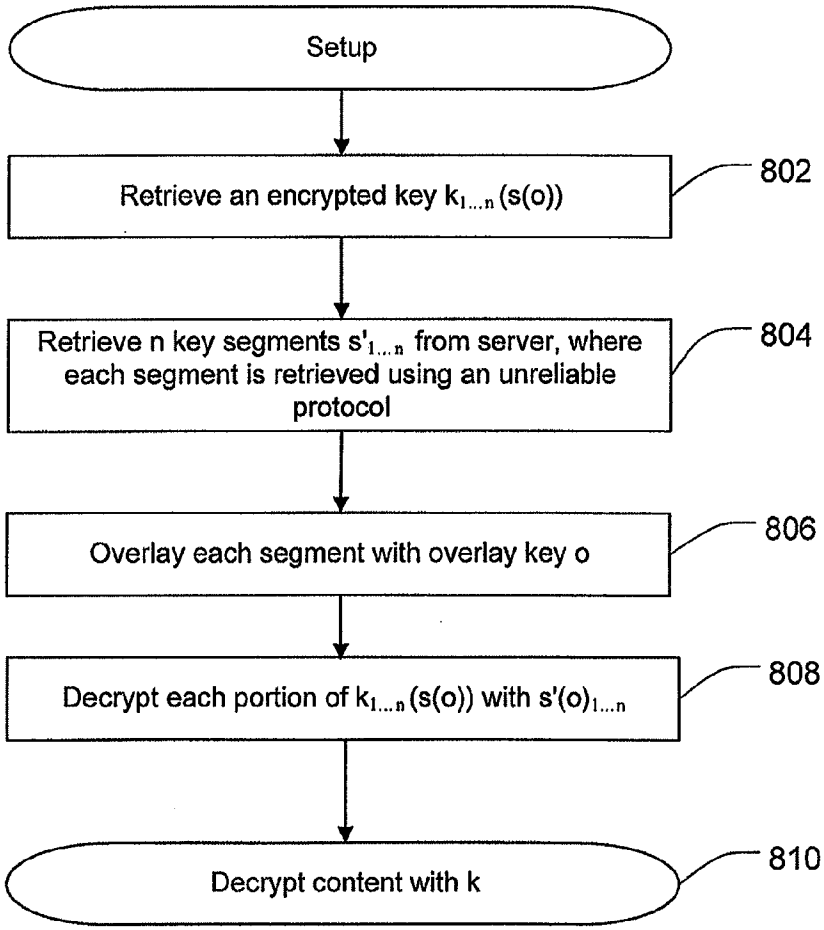
Setup
Retrieve an encrypted key $k_{1...n}(s(o))$
802
Retrieve n key segments $s'_{1...n}$ from server, where each segment is retrieved using an unreliable protocol
804
Overlay each segment with overlay key o
806
Decrypt each portion of $k_{1...n}(s(o))$ with $s'(o)_{1...n}$
808
Decrypt content with k
810

FIG. 8

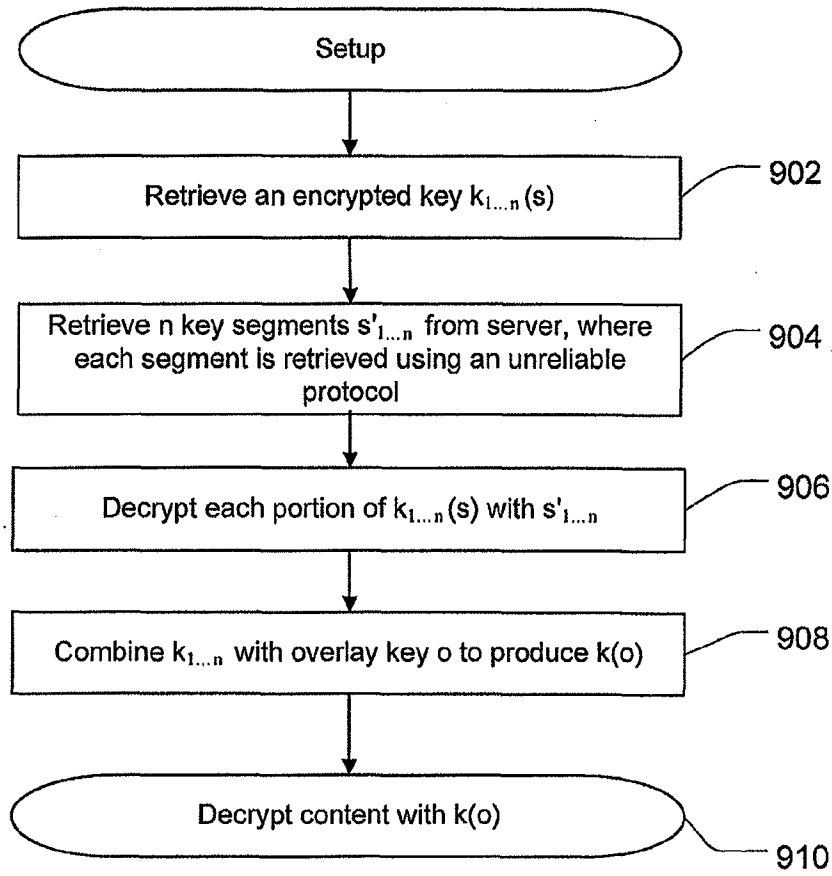
Setup
Retrieve an encrypted key $k_{1...n}(s)$
902
Retrieve n key segments $s'_{1...n}$ from server, where each segment is retrieved using an unreliable protocol
904
Decrypt each portion of $k_{1...n}(s)$ with $s'_{1...n}$
906
Combine $k_{1...n}$ with overlay key o to produce k(o)
908
Decrypt content with k(o)
910

FIG. 9

# INTERNATIONAL SEARCH REPORT

International application No
PCT/US2012/024400

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04N21/4405 H04N21/4408 H04N21/433 H04N21/835 H04N21/6334
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| Y | JIEYIN CHENG, CHEUN NGEN CHONG, JEROEN DOUMEN, SANDRO ETALLE,PIETER H HARTELAND STEFAN NIKOLAUS: "STREAMTO: STREAMING CONTENTUSING A TAMPER-RESISTANT TOKEN", , [Online] 12 December 2005 (2005-12-12), XP002680894, Retrieved from the Internet: URL:http://doc.utwente.nl/56985/1/0000011c .pdf> [retrieved on 2012-03-30] page 6 - page 8 page 11 figure 6 ----- -/-- | 1-25 |

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

| Date of the actual completion of the international search | Date of mailing of the international search report |
|---|---|
| 27 July 2012 | 08/08/2012 |

| Name and mailing address of the ISA/ | Authorized officer |
|---|---|
| European Patent Office, P.B. 5818 Patentlaan 2<br>NL - 2280 HV Rijswijk<br>Tel. (+31-70) 340-2040,<br>Fax: (+31-70) 340-3016 | Fantini, Federico |

2

Form PCT/ISA/210 (second sheet) (April 2005)

# INTERNATIONAL SEARCH REPORT

International application No
PCT/US2012/024400

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| Y | WO 2007/146763 A2 (SCIENTIFIC ATLANTA [US]; PINDER HOWARD G [US]; MAHOLSKI ANDREW D [US])<br>21 December 2007 (2007-12-21)<br>abstract<br>page 1, line 7 - page 2, line 5<br>page 2, line 19 - line 23<br>page 5, line 1 - line 5<br>page 7, line 23 - page 8, line 4<br>page 9, line 14 - line 16<br>page 11, line 9 - line 16<br>page 14, line 7 - line 16<br>page 14, line 22 - page 15, line 18<br>page 18, line 13 - line 23<br>page 20, line 1 - line 5<br>page 22, line 22 - page 23, line 7<br>page 23, line 15 - page 24, line 9<br>page 24, line 15 - page 25, line 6<br>page 27, line 11 - page 28, line 11<br>figures 7,8,9<br>----- | 1-25 |
| A | JUNGKYU HAN ET AL: "EMCEM: An Efficient Multimedia Content Encryption Scheme for Mobile Handheld Devices",<br>INFORMATION SCIENCE AND SECURITY, 2008. ICISS. INTERNATIONAL CONFERENC E ON, IEEE, PI,<br>1 January 2008 (2008-01-01), pages 108-114, XP031207429,<br>ISBN: 978-0-7695-3080-X<br>abstract<br>sections 1, 2.2, 2.2.2, 3.2;<br>page 108 - page 111<br>----- | 1-25 |
| A | EP 2 249 254 A2 (SANDISK IL LTD [IL])<br>10 November 2010 (2010-11-10)<br>abstract<br>paragraph [0001]<br>paragraph [0007] - paragraph [0011]<br>paragraph [0013]<br>paragraph [0023]<br>paragraph [0037]<br>paragraph [0084]<br>paragraph [0094] - paragraph [0095]<br>paragraph [0102] - paragraph [0103]<br>paragraph [0107]<br>paragraph [0134] - paragraph [0136]<br>paragraph [0147]<br>paragraph [0155]<br>paragraph [0157] - paragraph [0161]<br>-----<br>-/-- | 1-25 |

2

# INTERNATIONAL SEARCH REPORT

International application No
PCT/US2012/024400

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| A | US 2006/129496 A1 (CHOW RICHARD T [US] ET AL) 15 June 2006 (2006-06-15)<br>abstract<br>paragraph [0003]<br>paragraph [0017]<br>paragraph [0022]<br>paragraph [0025]<br>paragraph [0037]<br>paragraph [0039]<br>paragraph [0048] - paragraph [0051]<br>paragraph [0053] - paragraph [0054]<br>paragraph [0057] - paragraph [0058]<br>figure 7<br>----- | 1-25 |
| A | WO 2009/105280 A2 (SECURITY FIRST CORP [US]; BONO STEPHEN C [US]; GREEN MATTHEW D [US]; L) 27 August 2009 (2009-08-27)<br>abstract<br>paragraph [0012] - paragraph [0013]<br>paragraph [0015]<br>paragraph [0059]<br>paragraph [0090]<br>paragraph [0092]<br>paragraph [0096]<br>paragraph [0099]<br>paragraph [0104]<br>paragraph [0107]<br>paragraph [0113] - paragraph [0121]<br>paragraph [0123]<br>figures 9A, 9B<br>----- | 1-25 |

2

# INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2012/024400

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| WO 2007146763 A2 | 21-12-2007 | CA 2655114 A1 | 21-12-2007 |
| | | EP 2052342 A2 | 29-04-2009 |
| | | EP 2375359 A2 | 12-10-2011 |
| | | KR 20090017604 A | 18-02-2009 |
| | | US 2007294178 A1 | 20-12-2007 |
| | | WO 2007146763 A2 | 21-12-2007 |
| EP 2249254 A2 | 10-11-2010 | CN 101960426 A | 26-01-2011 |
| | | CN 101976315 A | 16-02-2011 |
| | | CN 102012790 A | 13-04-2011 |
| | | EP 2243083 A2 | 27-10-2010 |
| | | EP 2249253 A1 | 10-11-2010 |
| | | EP 2249254 A2 | 10-11-2010 |
| | | JP 2011513804 A | 28-04-2011 |
| | | KR 20100106609 A | 01-10-2010 |
| | | KR 20100106610 A | 01-10-2010 |
| | | KR 20100107479 A | 05-10-2010 |
| | | TW 200937198 A | 01-09-2009 |
| | | WO 2009088709 A2 | 16-07-2009 |
| US 2006129496 A1 | 15-06-2006 | CN 101080724 A | 28-11-2007 |
| | | EP 1828951 A1 | 05-09-2007 |
| | | US 2006129496 A1 | 15-06-2006 |
| | | WO 2006065336 A1 | 22-06-2006 |
| WO 2009105280 A2 | 27-08-2009 | AU 2009215815 A1 | 27-08-2009 |
| | | CA 2716335 A1 | 27-08-2009 |
| | | CN 101981890 A | 23-02-2011 |
| | | EP 2163067 A2 | 17-03-2010 |
| | | EP 2416541 A1 | 08-02-2012 |
| | | US 2009254750 A1 | 08-10-2009 |
| | | WO 2009105280 A2 | 27-08-2009 |