RDK + CA/DRM: ENHANCING INTEROPERABILITY FOR ROBUST REVENUE SECURITY

Petr Peterka
Verimatrix

Abstract

Open source platforms such as RDK offer many advantages, including the benefit of having a common implementation instead of a common specification; however, they are not without challenges, such as security threats and interoperability with proprietary components such as conditional access system (CAS) or digital rights management (DRM).

In this paper, the author explores how to optimize CA/DRM integration in the RDK bundle and ensure the revenue security of RDK applications.

The pay-TV industry relies on many moving parts to put an end-to-end television distribution system together. The set-top box (STB) on its own is an expensive custom-designed piece of consumer electronics. The Reference Design Kit (RDK) is trying to ease the cost burden of designing and integrating the STB by providing a common reusable middleware platform.

As a joint venture between several operators (Comcast Cable, Time Warner Cable, and Liberty Global), the RDK open source platform is well-poised for future growth and adoption. The platform currently has more than 200 licensees but will certainly add more in coming months as it continues to expand its capabilities. Although RDK was initially designed for the needs of North American operators, it is becoming allocable world-wide. Consider for example RDK Management's recent announcement that RDK now features support for the Digital Video Broadcasting (DVB) standard[1], which is widely used throughout Europe and other parts of the world.

UNDERSTANDING THE BENEFITS

RDK has become an increasingly popular pre-integrated software bundle as it offers many advantages, including the ability to support a common implementation instead of a common specification, which can accelerate the deployment of next-gen video products and services. According to RDK Central, using the bundle can also "enable TV service providers to standardize certain elements of these devices, but also to easily customize the applications and user experiences that ride on top."[2]

There are also cost benefits – RDK is a $0/royalty free commercial source code license – as well as the ability to enhance collaboration through full transparency into the source code. As an open source code initiative, the RDK enables community members to add new features and resolve issues on their own schedule and make contributions back to the community. Such an approach enables multichannel video programming distributors (MVPDs) to focus, innovate, and differentiate at the application and

[1] http://rdkcentral.com/rdk-management-expands-dvb-support-for-operators/
[2] www.redkcentral.com

services layer.

OUTLINING THE CHALLENGES

However, RDK is not without challenges, such as security threats and interoperability with proprietary components such as conditional access system (CAS) or digital rights management (DRM). Content protection in the form of a CAS client and DRM client must be supported by RDK in order to realize its benefit, however today's proprietary integrations with CAS clients are not scalable and create forked versions that are difficult to keep in sync with the main trunk RDK code. Once a forked version becomes stale and starts missing new features that have been developed after the forked version, the RDK benefits quickly diminish.

It is also worth noting that while RDK takes advantage of the open-source nature of RDK-M managed software development, the OSS license obligations need to be carefully managed when integrating with a proprietary implementation of a CAS or DRM client. Specifically, the abstract CAS APIs integrated into RDK must have an OSS license that allows CAS or DRM clients to be plugged in without putting OSS obligations on the CAS vendor that they could not comply with.

RDK AND REVENUE SECURITY

CA/DRM plays an important role in an RDK bundle. Typically, an MSO is providing a pay-TV service and needs to protect that service and the associated revenue. While, subscribers are mainly interested in premium content such as Disney, HBO or ESPN, content owners are focused on protecting their content from theft and illegal distribution. This is why MSOs have used CAS in their content distribution networks for decades.

Even though a CAS client is typically a very small component for an RDK bundle, it interacts with several other modules within RDK as well as the overall device such as a set top box (STB), DVR, HDMI dongle or a connected TV. There are several main interfaces:

- **Interaction with the content tuner/demultiplexer** – typical liner and on-demand television channel is encoded in MPEG-2 Transport Stream (TS) format. Systems-on-chips (SOCs) and middleware such as RDK provide tuning and de-multiplexing capability to optimize the channel tuning and channel changing experience. When a channel is encrypted by a CAS system, the TS carries additional messages such as EMMs and optionally EMMs. These need to be provided to the CAS client in order to properly decrypt the content. Newer systems utilize other formats such as MPEG-DASH, which has an equivalent mechanism to carry CAS or DRM specific data.

- **Interaction with a hardware root of trust** – most premium content requires high level of security robustness. It can be achieved through several methods but most often the system relies on unique hardware identity and hardware-based keys. Such functionality can be exposed through a standardized key ladder or Trusted Execution Environment (TEE). Additionally, content is typically decrypted in hardware for performance and security reasons; therefore, additional access to these hardware resources is required.

- **Interaction with the application layer –** it is desirable to isolate the CAS functionality from the application layer as much as possible but some high-level APIs are needed. (This will be addressed in more detail later in the paper.)

- **Access to IP stack** - as many operators are migrating to all-IP architectures, IP-based CAS systems are more suitable than traditional 1-way systems. IPTV CAS utilizes the 2-way IP connectivity back to the MSO head-end in order to perform device authentication, entitlement and key retrieval at the time protected content is being accessed.

- **Forensic watermarking –** premium content such as UHD, requires additional level of protection in the form of forensic watermarking. Several methods of embedding the mark exist. Technologies embedding a mark into the decoded video require access to the decoder.

The CAS/DRM cannot be included in the open source as CAS vendors require a strict control over the CAS client implementation to guarantee integrity and authenticity. Furthermore, intellectual property included in the secure key management protocol and other cryptographic algorithms needs to be protected. If CAS client code was included in the open source, hackers could include “back doors” or gain too much knowledge about the system which could help them in defeating overall security.

Ideally, the CAS client should be isolated from the middleware and application functionality and minimally intrusive on the rest of the functionality. Therefore, abstracting out the main functions without exposing unnecessary details of the CAS system is highly desirable.

Furthermore, the MSO may want to retain the choice of with which CAS vendors they want to partner. A well-designed abstraction layer allows them to relatively easily upgrade one CAS client version with another or even replace one CAS technology for another.

MITIGATING THREATS TO REVENUE SECURITY

Isolating the CAS functionality also helps eliminate any negative impact on the RDK functionality, evolution and innovation while preserving the ability to quickly integrate a CAS system of choice without proprietary integration into RDK or requiring special forked versions of RDK. At the same time, any CAS client that will be integrated with RDK can be properly isolated through a set of abstract APIs whose OSS license allows a plug-in of a proprietary CAS client without contaminating it with undesirable OSS obligations.

Even though security and content protection shouldn't be exposed to the user in order to keep the user interface as simple as possible, there are some activities that need an interaction between the application and the CA client.

Tuning/Navigation

Tuning, channel change or on-demand asset selection are the main activities initiated by the user. When a protected channel or asset is selected, the proper entitlement and decryption must occur. When everything works well, the user shouldn't even see the security activity that is triggered in the background. The main steps shown on the diagram below (as application use case #1) are originated by the application (e.g. EPG, VOD library, PVR playback) and request the tuning (or more generally content selection) module which tunes to the indicated channel, starts parsing the associated metadata and determines whether this content needs any interaction with the CAS subsystem at all. Content metadata (e.g. MPEG PMT tables) may indicate presence of CAS or DRM information (e.g. ECMs, DASH PSSH). This is a point of interaction with the CAS client as this information needs to be passed in for CAS processing. Once the CAS function completes, assuming that the user is authorized to watch the selected content, the decryption module is properly armed.

Of course, behind the scene, the CAS client will use the TCP/IP stack to communicate with key management servers in the head-end to obtain content decryption keys, and with the SOC security functions to achieve desired level of robustness.

The scenario is very similar between tuning to a live channel, playing from a local or network DVR, or selecting a movie or TV episode from a VOD library.

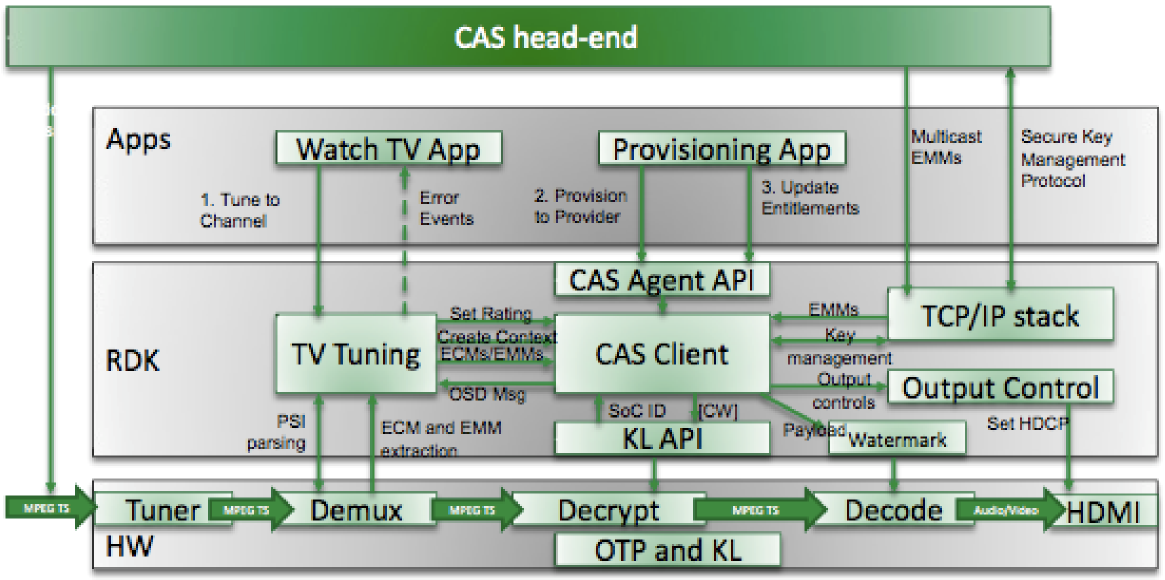
Initialization/Provisioning

STBs or more generally user devices capable of consuming MSO-originated content may be leased by the operator following the typical traditional model, or could be retail devices such as connected TVs, retail STBs or dongles, etc. In the leased model, the STB and the CAS client may be preconfigured to connect and register to the operator with very little or no interaction with the user or the application. A retail device may be connected to one of several MSOs without the device manufacturer knowing the specifics. In this case, it may be the application that provides configuration information to the CAS client to provision to a specific MSO key management or license server.

This is shown on the diagram below as application use case #2.

Entitlements

CAS systems have different ways for obtaining entitlements. In traditional systems they are delivered using EMM messages that are demultiplexed from the MPEG 2 Transport Stream and passed to the CAS client. In IP-based systems, the CAS client contacts the head-end key management server and requests entitlements (and corresponding keys) over the 2-way IP channel. Most of this functionality can be hidden from the application but under certain circumstances, the application may trigger an entitlement refresh action. The reason may be that the user subscribed to a new channel but the CAS client has not proactively requested the corresponding keys, yet. This is shown on the diagram below as application use case #3.



On-screen Display (OSD)

Even though CAS clients typically do not have a user interface, there are scenarios where it needs to provide some status to the user or even solicit input form the user.

For instance, when a user tunes to a channel that it is not entitled for, the decryption and eventual rendering of the content will fail. An application should know the reason so that it can provide adequate information to the user, show instructions how to sign up for that channel or tune to an alternative channel.

Another example is based on events that may be controlled by the user such as parental control or pay-per-view purchases. If these events require a user action, such as providing a security PIN code, the CAS client must have an ability to communicate with the right application.

When a specific channel or on-demand content is forensically watermarked, the use may need to be notified. This is another instance where the CAS client needs to provide additional information to the user via the application layer.

THE POWER OF STANDARDIZATION

There are different methods to standardizing on RDK. One way is to create an OSS implementation and share it with all vendors who need that functionality. All users of OSS benefit from a single implementation centralize code management process and testing responsibilities. Another, possibly more traditional method is to create a standards organization which publishes a specification based on contribution of interested parties.

There are several standards organizations that help the TV industry. The main international one is MPEG but there are several others at the national level, including SCTE, CEA, CableLabs, etc.

One area that has been historically covered by a proprietary integration between a SOC vendor and the CAS vendor is what is generically called hardware root of trust. This may include unique chip identifier, access to unique chip root keys, a proprietary key ladder which implements a complex key hierarchy that is specific to each CAS vendor, and actual content decryption by secure hardware. There has been progress made in the last several years to standardize this functionality in order to assist with CAS interoperability. Such approaches make client downloadability more feasible, provide options for running multiple CAS clients on the same device or to replace an old version of a CAS client with new one.

An example of such standards is ETSI OMS KLAD[3] and its North American extension SCTE 201[4]. This allows chip vendors to manufacture CAS-independent SOCs that can be used in devices that need the ability to download a CAS client either during the device manufacturing process or later when the device is installed in consumer's home. Such a solution may also require a Trust Authority (TA) that generates and manages the chip root keys and provides them to all CAS vendors who end up using those chips. Such companies exist today and provide this role to the industry today.

Another standard that is very beneficial to the CAS interoperability goal is DVB Simulcrypt[5]. When an operator runs their pay-TV distribution system in the Simulcrypt mode, two or more CAS systems can coexist on the network. Some devices integrate or download CAS from vendor A while other devices may integrate a solution from CAS vendor B. Such systems have been successfully deployed around the world.

These standards when supported by RDK will make integration with CAS clients much faster and almost plug-and-play.

When an in-the-field or over-the-air downloadability of a specific CAS client is required, the actual download mechanism and the run-time environment need to be standardized as well. This is a very complex and difficult task but good progress has been done by the industry especially by GlobalPlatform[6]. The most relevant aspect of the GP activities is related to Trusted Execution Environment (TEE). The TEE architecture splits the SOC execution into a trusted world and untrusted world. Security sensitive

[3]http://www.etsi.org/deliver/etsi_ts/103100_103199/103162/01.01.01_60/ts_103162v010101p.pdf

[4]http://www.scte.org/documents/pdf/Standards/ANSI_SCTE%20201%202013.pdf

[5]http://www.etsi.org/deliver/etsi_ts/103100_103199/103197/01.05.01_60/ts_103197v010501p.pdf

[6] https://www.globalplatform.org/mediaguidetee.asp

computation such as ECM processing, content decryption and content watermarking is performed by a trusted application (TA) running in the TEE while non-secure parts of the CAS client run in the rich execution environment (REE).

The REE part would primarily implement the mapping of the Abstract CAS API, which is used by the RDK internal modules to communicate with a CAS client, to the specific implementation of the CAS client running as a TA in the TEE. By securely downloading a new TA, the CAS functionality can be upgraded or even replaced by a different implementation. GlobalPlatform standardizes the APIs for communicating between the REE and the TEE as well as access to cryptographic functions inside the TEE to make portability of TAs between SOCs as transparent as possible.

TEE has its own security architecture including secure boot, TA authentication and decryption, as well as HW root of trust. Additionally, the TEE may provide access to the OMS key ladder discussed above.

LOOKING FORWARD

From what has been discussed so far, it is clear that a significant amount of progress has been made already, RDK with all its OSS components is stable and enjoys real deployments. Standards such as SCTE, GlobalPlatform, MPEG-DASH have released stable versions of their specifications. What is still missing are the abstract CAS APIs and HW security abstraction.

The HW security abstraction can be modelled after the SCTE OMS standard and expose HW functions such as “get chip identifier”, “host authentication”, “set descrambler” and “get random number”.

Tuner and demultiplexer modules of RDK will need APIs to “create a tuning session” such that multiple channels can be processed simultaneously. Within each context, CAS specific data, such as ECMs and EMMs, will be passed to the CAS client via “send CAS data” API, and finally the descrambling will be activated/deactivated by “start session” and “stop session”, respectively.

Application oriented APIs will include “CAS initialization” to complete proper provisioning into a service, “purchase asset” and “update entitlements” related to access VOD or linear channels, “set parental rating” to limit what can be watched, and “get message” for any information that needs to be communicated to the user.

The detailed object model of these APIs is certainly more complex that the outline above but several companies spent significant amount of time to put a concrete proposal together that is being evaluated by the RDK community.

It is not difficult to see how the three seemingly antagonistic worlds of open source software, international and national standardization, and proprietary content protection can come together to assist in rapid deployment of a complex ecosystem.