API for Encryption Key Exchange

Version	0.6
Distribution	Informal Key Exchange Working Group
Date	Jan 4, 2011

Revision History

Date	Revision	Editor	Comments
Sep 14, 2010	0.1	YF	Initial document
Nov 17, 2010	0.2	SRC	Elaborated on context, terminology, requirements
Nov 30	0.3	YF	Added comments from Envivio, Azuki, Verimatrix
Dec 20	0.5	YF	Comments from Harmonic and call on Dec 14
Jan 4, 2011	0.6	YF	Accept revisions and clarify text

Table of Contents

1 Overview

Delivery of high-value video content requires encryption and careful management of the associated keys. Encryption keys are typically managed by CA/DRM systems that incorporate specific key-management servers (KMS), designed to distribute keys to client devices for the decryption of content in a controlled fashion.

In many situations, the Scrambler component that encrypts the stream is not part of the KMS, but must use encryption keys that are ultimately managed by the KMS. Moreover, often the KMS is built by a DRM vendor while Scramblers are built by encoder vendors. Unfortunately, the exchange of keys between the KMS and scrambler to keep the two components in synch is not standardized and each vendor typically uses proprietary interfaces.

The simplest case of Scrambler integration with a KMS is for the preparation of a single bit rate protected on-demand file. More complex scenarios today involve the real-time encryption of adaptive bitrate streaming services, where content is split into segments or chunks representing a specific time period encoded at one or more compression rates. In this scenario encryption keys get associated with the specific time periods of the content, or perhaps even the different bit rates of each stream. Also, with any real-time service, the issue of redundancy management must be addressed.

The goal of this document is to list requirements for an API for exchanging keys between key management servers and scramblers that can scale from the simplest situations to the most complex and demanding that are envisioned. Here, a ‘key management server’ is a component in the control path that creates encryption keys as required for specific streams or content files, where the content is organized within a specific identification scheme. [NOTE: Should we be thinking about a Simulcrypt solution. Today this is supported for example in the Microsoft PIFF 1.1 format. In this case it may be more sensible for the Scrambler to generate the key as in DVB simulcrypt.] The KMS distributes the keys to components, more in the data path, that encrypt files or streams and subsequently manages distribution of keys to end-clients that receive the encrypted content. The ‘scrambler’ is a component that encrypts the streams using some sort of encryption key. The identification scheme enables the client devices to relate the encrypted content to the required key for decryption.

This document does not cover any aspect of the interaction between the KMS and the client device or specify how the content identification scheme must be organized.

2 Terms

Key Management Server (KMS) – a component that creates encryption keys and manages access and distribution of those keys between various other components in a video delivery chain.

Scrambler - is a component that encrypts the streams using some sort of encryption key.

Segment - is some time period of content within a larger file or stream, where the encoder has created a self contained set of information. Also sometimes termed a chunk.

Content ID - Identifier of a specific stream or file as understood by the KMS. Can be combined with a time reference for real-time use.

Encryption key - A value used within a scrambling algorithm when encrypting a file or segment of a file/stream.

Initialization Vector - An additional seed parameter required by some types of encryption algorithm.

3 References

TBD - anything that already impacts a specific aspect of the standard

4 Scope

The API shall not be limited to one technology, use-case, or vendor. The API should be applicable in any use-cases in which a key-exchange mechanism is required.

5 Requirements

We define requirements for each component

5.1 General Protocol Requirements

5.1.1 The API between the KMS and Scrambler should be openly published. [NOTE: We should endeavor to have RAND licensing terms.]

5.1.2 The key exchange protocol shall be designed to be stateless in nature, except for state mechanisms (such as cookies) that are part of the underlying communication protocol.

5.1.3 The key exchange protocol shall support redundant configurations of scramblers and KMS devices.

5.1.4 The scrambler shall initiate the session, initiate requests for keys, determine interval of key requests, manages which KMS is used for primary and redundant delivery, and determine the content identifier and time stamp used in each key request

5.2 KMS Requirements

5.2.1 The KMS shall be capable of creating encryption keys on demand

5.2.2 The KMS shall be capable of storing encryption keys together with an associated content ID reference

5.2.3 The KMS shall be able to create keys for both VoD and live content.

5.2.4 The KMS may support rotating keys - that is keys that are renewed on a periodic basic during a streaming of live content

5.2.5 The KMS shall be able to signal which types of encryption algorithm and keys it supports. [Note, for example, that the same key might be used for both PlayReady and HLS-based AES, so that the encryption can happen once with the same key, even if the delivery protocol for the content and keys may differ.]

5.3 Scrambler Requirements

5.3.1 The scrambler shall be able to communicate and accept keys from at least two distinct KMS service points, defaulting to the second KMS if the first fails after a pre-determined time-out.

5.3.2 The Scrambler may be able to create Encryption keys. [Note: this may facilitate the use-case in which the same key is used for multiple delivery protocols.]

5.4 Type of Data Exchanged

5.4.1 The encryption keys shall be able to be represented as a character string

5.4.2 The content identifiers and timestamps shall be able to be represented as a character string

5.4.3 Data exchanged between KMS and scrambler shall be sufficient to enable unambiguous key requests from client devices for the scrambling keys used at each point in a media file or on a media stream

5.4.4 The API should support exchange of auxiliary data, such as ratings data and output protection data.

5.4.5 The API should allow for the insertion of all data necessary for decryption. [Note: In the DVB simulcrypt example, this would be the Super CAS ID; for Microsoft PIFF, this would be the Protection System Specific Header Box.]

5.5 I/O

5.5.1 The key exchange mechanism shall take place using HTTP, in the clear or using SSL to protect the data exchange

5.6 Authentication

5.6.1 The scrambler and KMS shall be able to authenticate each other prior to any key exchange.

5.6.2 The Scrambler and KMS shall announce their supported authentication schemes and negotiate a mutual scheme used to authenticate each other.