3GPP TSG SA WG3 Security — S3#31 **S3-030805**
18 - 21 November 2003
Munich, Germany

Title:	**LS on Protection of MBMS and DRM Streaming Services**
Response to:	**S3-030756 and S3-030758 (Source: Download+DRM group of the Open Mobile Alliance)**
Release:	**Rel-6**
Source:	**SA3**
To:	**SA4, OMA DLDRM, SAGE**
CC:	**SA1,**

Contact Person:

Name:	Krister Boman
Tel. Number:	+46 31 747 4055
E-mail Address:	krister.boman@ericsson.com

Attachments: **S3-030750, S3-030756, S3-030752**

Overall Description:

3GPP TSG WG SA3 would like to thank OMA DLDRM for their LSs and would like to provide with the following comments and guidelines to the LS SA3-030756.

SA3 has agreed upon the following guidelines when developing the security protocols for protection of MBMS and DRM Services:

- A harmonization between the DRM and MBMS security is a key issue in order to alleviate the compelling negative impact on the terminal should different security protocols be chosen for DRM and MBMS streaming services.
- SA3 has thus far mainly discussed the use of SRTP as a candidate protocol for protection of MBMS streaming services however SA3 has not yet decided to implement SRTP in the MBMS specifications yet since it is dependant on decisions in SA4 on codecs and in OMA DLDRM group for DRM protection. SA3 has noted that SRTP is now approved by IETF/IESG for publication, but it is also noted that the proposed deviations from SRTP specific transform in S3-030750 like e.g. the IV construction have not yet been evaluated by SA3.
- SA3 would like to point out that it is a key issue for SA3 to re-use cryptographic algorithms and security protocols that have undergone a public review and been scrutinized during a longer period of time.
- SA3 would also like to highlight that the key management for DRM and MBMS user services may be independent. There is currently an initiative to develop MBMS specific key management procedures for MBMS user services not utilizing DRM services. It is therefore a key issue for SA3 to develop a complete solution that integrates, user authentication, key management with the protection of RTP traffic for MBMS services. SA3 has also discussed the contribution S3-030752 which proposed principles by which the SA3 work on MBMS security can be aligned with the ongoing co-operation between OMA DRM group and SA4 group for PSS but no decisions could be made.
- SA3 assumes that the proposal from DLDRM is compliant with the requirements of OMA as highlighted in the LS from DRLDRM, S3-030756. SA3 wants to highlight that there is an integrity protection requirement which is optional for use for MBMS. This particular requirement assumes that the user is trusted, while in the general MBMS trust model, it is assumed that the user is untrusted. SA3 also wants to point out that the selective encryption scheme may not be compliant with the current following requirements of MBMS in the draft TS 33.246:
  - o *R6a: It shall be possible to protect against unauthorized modification, insertion, replay or deletion of MBMS multicast data sent to the UE on the radio interface. The use of integrity shall be optional.*
  - o *R7a: It shall be possible to protect the confidentiality of MBMS multicast data on the radio interface.*
  - o *R7d: It shall be infeasible for a man-in-the-middle to bid down the confidentiality protection used on MBMS multicast session from the BM-SC to the UE.*
  - o *R7e: It shall be infeasible for an eavesdropper to break the confidentiality protection of the MBMS multicast session when it is applied.*

Furthermore, SA3 finds AES in Counter Mode with 128 bit key acceptable for cipher suite as proposed in S3-030750 and S3-030756. There were comments raised that a review from SAGE would be suitable in order to verify that this is a feasible approach from a cryptographic point of view.

Actions:

To OMA DLDRM group

ACTION: 3GPP TSG WG3 asks OMA DLDRM to

- **Consider the guidelines and the key issues identified for MBMS service above when further progressing the security for DRM services**
- **SA3 asks further feedback from OMA DLDRM on the progress for the security of DRM services since it is related to future decisions in SA3 on securing MBMS services**

To : 3GPP TSG SA4

ACTION: 3GPP TSG WG3 asks 3GPP TSG SA4 to

- **Give feedback to TSG SA3 whether the SRTP transform as proposed in S3-030750 is a suitable and feasible mechanism for securing MBMS streaming services from an SA4 point of view**

To : ETSI SAGE

ACTION: 3GPP TSG WG3 asks ETSI SAGE to

- **Review the AES CTR mode proposals from a cryptographic point of view and comment on its suitability for protecting content**
- **If possible comment on the selective encryption mechanism in particular considering the references to research papers in S3-030750**

Date of Next SA3 Meetings:

SA3#32	9 – 13 February 2004	TBC, EF3
SA3#33	11 - 14 May 2004	Beijing, Samsung
SA3#34	6- 9 July 2004	NN
SA3#35	5 – 8 October 2004	Sophia Antipolis, EF3 (TBC)

3GPP TSG SA WG3 Security — S3#31 **S3-030750**
18 - 21 November 2003
Munich, Germany

Source:	**Ericsson**
Title:	**Considerations on selective encryption and integrity protection for DRM protected PSS media streams**
Document for:	**Discussion**

1 Introduction

3GPP has delegated the standardization of DRM [1] to OMA [2]. However, it turned out that for interoperability of 3GPP PSS streaming and 3GPP MBMS with OMA DRM 2.0, adaptations on both ends are necessary. OMA has proposed that 3GPP defines the protected file format and the streaming mechanisms for protected PSS media [3][4].
In OMA, the use of selective encryption of streams was supported by a majority of companies. Further, the issue of stream integrity protection has been discussed. Although there were companies that proposed the use of stream integrity protection, and although the OMA DRM group has included integrity protection for downloadable content in their spec draft, it was concluded that stream integrity protection is not a DRM requirement per se [5]. However, the OMA DRM group acknowledged in the recent LS that SA3 and SA4 may have further considerations, and left the final decision on stream integrity protection to 3GPP [5].

Ericsson believes that OMA DRM has not sufficiently considered privacy and security threats that are introduced through selective encryption and the combination of selective encryption without integrity protection. We outline these threats here and propose a solution to address the threats.

2 Why integrity protection of PSS streams is required

The main problem is the use of selective / partial encryption which has been proposed by several companies. The idea is that individual (RTP) packets of DRM protected streams can be encrypted or not, and that this is signaled by a ‘flag’ within the respective packet. The main argument for selective encryption is savings in computational complexity. However, the vulnerabilities and resulting security threats that are introduced have not sufficiently been addressed.

The following vulnerability A. is introduced by the use of selective encryption:

<u>A. Streams that are only partially encrypted can be reconstructed with sufficient quality</u>

The usual argumentation is that essential parts of a video or audio stream are protected, such that the unencrypted parts are ‘useless’ and cannot be used to reconstruct the stream. Research results have shown that this assumption is dubious from a security and privacy point of view. Even if the stream cannot be reconstructed with full or good quality, thus diminishing the business value, it can often be reconstructed well enough to determine what content it contains. Agi and Gong [8] selectively encrypted video clips and were still able to recognize what type of scenery was contained in the sequence. They state "...In this paper we have reported an empirical study of MPEG video encryption. We found that these methods are not adequate for sensitive applications. Specifically, our experiments confirmed our intuition that encrypting I-frames alone may not be sufficiently secure for some types of video...". Similar observations were made by Lookabaugh et al. [10] who say "... Our particular evaluation of selective encryption schemes for a “neutral” relationship between compressor and encryptor shows that the system is not particularly robust against reasonable statistical and perceptual attacks if we target a low percentage of selective encryption by focusing on headers. ...", and Zeng et al.[11]: “Depending on how significant the impact [of the selective encryption][…] on the visual quality, and on how predictable/recoverable the [encrypted portions][…] are based on other unencrypted data, the resultant encrypted bitstreams may have different levels of security.”. Even the paper by Wen et al. [12] which supports selective encryption in general states that “…encrypted multimedia content is subject to error concealment based attacks, which are based on trying to conceal the unbreakable encrypted data based on other available data.”

Although selective encryption may be sufficient to diminish the quality of video streams, it is not sufficient to prevent eavesdroppers from at least understanding what the video is about, thus imposing a potentially very serious privacy vulnerability, and possibly even reconstructing a low-quality version of the video.

Moreover, the gain through selective encryption is not significant; Li, Zhang, Tan, Campbell [9] found that the encryption of I frames only decreased the decoding speed in terms of frames per second of their reference decoder by 11-16 %, encryption of all frames by 14-23 %.

The following vulnerabilities B. and C. are introduced by the combination of “selective encryption” and “no integrity protection”:

B. A man-in-the-middle or the legitimate receiver can manipulate the stream

If selective encryption is used on a packet-per-packet basis, and is signaled in the packet itself, a man-in-the-middle (or the legitimate user) could replace each unprotected (no encryption/no integrity protection) packet by any other packet. Further, he could replace protected packets by unprotected packets with arbitrary content. Thus, a man-in-the-middle could manipulate or damage the content and the legitimate receiver had no means to detect that this is not the version as sent by the streaming server; this would impair the credibility of the streaming server/content provider

Even if there was integrity protection, but just on the (RTP) payload, and not on (RTP) packet headers including packet number and timestamp, packets could be exchanged or replayed. Thus, a man-in-the-middle could reassemble the video stream and e.g. exchange the order of scenes, by just changing the packet order and adapting the packet headers accordingly. This can be done even for encrypted packets, if the decryption does not depend on previous packets (as it typically does in environments with significant packet loss probability). In case RTCP feedback is used for streaming services, it can also be manipulated if it is not integrity protected.

C. “Selective encryption off” must be signaled securely

Even if selective encryption is not used for a whole particular stream, this must be signaled securely. Otherwise a man-in-the-middle can intercept this information and set to “selective encryption on”, and can replace all protected packets by arbitrary other unprotected packets. The secure signaling of DRM information is in general advisable; for example also integrity protection of the URL pointing to the rights issuer that issues rights objects for a stream. Otherwise, this information could be replaced by a man-in-the-middle.

3 Proposal

Summarizing, although selective encryption and missing integrity protection do not lead to leaking of protected content, which is the main DRM concern, they lead to other unacceptable vulnerabilities and threats.

1. In order to avoid the vulnerabilities outlined in the previous section, Ericsson proposes that 3GPP SA3 decides for the following:

   (A) 3GPP SA3 and SA4 do not specify or allow selective encryption for DRM protected PSS streams[1]

   (B) 3GPP SA3 and SA4 specify a mechanism for integrity protection of DRM protected PSS streams (mandatory to implement on PSS-DRM servers and clients, optional to use) that integrity protects payload and packet headers

2. The Secure Real-Time Transport Protocol (SRTP) [6][7] is one possible method for integrity protection of streams and has undergone security considerations in IETF. Ericsson suggests considering SRTP as a mechanism for stream integrity protection.

4 References

[1] 3GPP TS 22.242 v2.0.0 (DRM Stage 1 document), ftp://ftp.3gpp.org/tsg_sa/WG1_Serv/TSGS1_16_Victoria/Output/S1-021185.zip

[2] LS on Digital Rights Management (from 3GPP SA to OMA), September 2002, see http://www.3gpp.org/ftp/tsg_sa/TSG_SA/TSGS_17/Docs/PDF/SP-020626.pdf

[3] OMA-DLDRM-2003-0081R01-3GPP-SA4-liaison, “Liaison on DRM content format from OMA DLDRM to 3GPP SA4”

[4] OMA-MAG-DLDRM-2003-0172R1-liaison-to-3GPP-SA4, ” Liaison to 3GPP SA4”

[5] OMA-BAC-DLDRM-2003-0221R3-liaison-to-3GPP-SA4-and-SA3, “Liaison to 3GPP SA4 and SA3 on issues on DRM for PSS and MBMS streams”

[6] SRTP, http://www.ietf.org/internet-drafts/draft-ietf-avt-srtp-09.txt

[7] Open-source implementation of SRTP, see http://srtp.sourceforge.net/srtp.html

[8] Iskender Agi and Li Gong, "An Empirical Study of Secure MPEG Video Transmissions", http://www.isoc.org/conferences/ndss96/agi.ps

[9] Li, Zhang, Tan, Campbell, "Security enhanced MPEG Player", http://choices.cs.uiuc.edu/Papers/Vosaic/se_mpeg_player.pdf

[10] T. Lookabaugh, I. Vedula, D. Sicker, "Selective Encryption and MPEG-2", http://itd.colorado.edu/lookabaugh/Documents/Selective%20Encryption%20and%20MPEG-2.pdf

[11] Wenjun Zeng, Jiangtao Wen and Mike Severa, “Fast Self-synchronous Content Scrambling by Spatially Shuffling Codewords of Compressed Bitstreams”, http://www.ee.princeton.edu/~wzeng/icip02_shuffling_preprint

[12] Jiangtao Wen, Mike Severa, Wenjun Zeng, Max Luttrell, and Weiyin Jin, “A Format-Compliant Configurable Encryption Framework For Access Control Of Multimedia”, http://www.ee.princeton.edu/~wzeng/mmsp01_PV_final_v2_preprint.pdf

[1] Should SA3 anyway decide to allow/specify selective encryption, we strongly recommend to follow the proposal (B) and to further specify a mechanism (mandatory to implement, optional to use) to integrity protect the information whether a stream is selectively encrypted or not. This information may e.g. be signalled in the SDP session description.

3GPP TSG-SA WG3 Meeting #xx
xxx, xxx, xxx yyy 2003

Tdoc S3 (03) 0xxx
Agenda Item: x.x.x

Source: **Ericsson**
Title: **Extensions to the 3GP file format for storage of encrypted / DRM protected media**
Document for: **Discussion**

1 Introduction

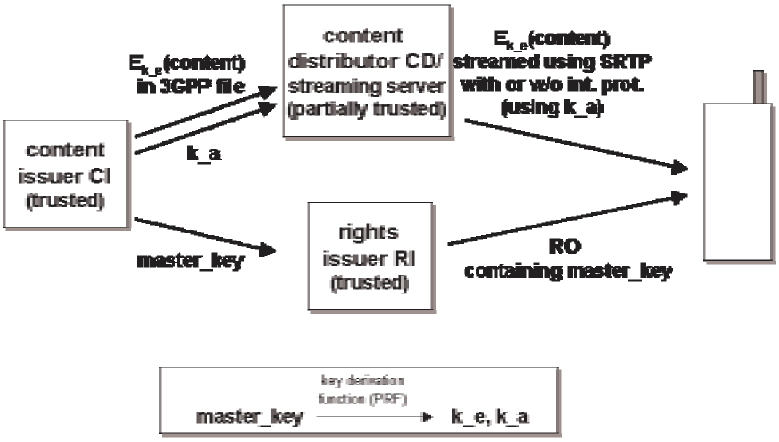
3GPP has delegated the standardization of DRM [1] to OMA [2]. However, it turned out that for interoperability of 3GPP PSS streaming and 3GPP MBMS with OMA DRM 2.0, adaptations on both ends are necessary. OMA has proposed that 3GPP defines the protected file format and the streaming mechanisms for protected PSS media [3][4], and key management is handled in the framework of the OMA DRM 2.0 specification. According to the requirements laid out in [3], media tracks are encrypted and stored in a 3GP file. The 3GP file can be downloaded as a whole, or encrypted packets can be extracted from the 3GP file and transported to the client using real-time transport protocols and mechanisms (that means transport protocols based on RTP/UDP).

This input proposes changes to the 3GP file format [6] that allow the storage and download of protected / DRM encrypted PSS media. The real-time streaming of protected media is not considered here and is subject of a separate input (Ericsson: Real-time transport of protected continuous PSS media).

2 Overview

Although this proposal and input (Ericsson: Real-time transport of protected continuous PSS media) do not depend on each other, they have been developed together. The basic idea is to encrypt content at the content provider site, store it in a 3GPP file and deliver it to a streaming server, and download or stream it from there.

For information, figure 1 shows the basic idea and the relation to the input (Ericsson: Real-time transport of protected continuous PSS media). The content provider uses a master key master_key. From the master key the integrity key k_a and content encryption key k_e (CEK in OMA terminology) can be derived using a key derivation function. The encryption is done at the content provider, and the encrypted streams stored in a 3GP file. The streaming server receives the encrypted content in the 3GP file and the integrity key k_a (if the CP/RI choose to apply integrity protection). The streaming server then streams the content to the client using SRTP. No additional encryption is applied. If applicable, the streaming server applies integrity protection. The content provider conveys the master key master_key to the rights issuer RI. The RI issues a rights object RO to the client, which contains the master key master_key. From master_key and knowing the key derivation function, the client can derive the content key k_e and (if applicable) the integrity key k_a. Subsequently, the client can decrypt the streams, check their integrity, and consume them according to the permissions contained in the RO.



3 File format extensions for storage of protected media

We propose to extend the 3GP file format with a mechanism for storage of encrypted media. The concept is expected to be standardised for the ISO base media file format by MPEG with 3GPP and ISMA in mind. In addition we define 3GPP-specific extensions that applies to encryption of text tracks and a 3GP profile brand for encrypted 3GP files. Details on the encryption scheme are stored in a protection information box. For the usage of encrypted 3GP files with OMA DRM 2.0, the exact details of the scheme will be defined by OMA.

The general idea behind the extensions is to replace code points (codec identifiers) of encrypted media with generic code points for encrypted media. This prevents legacy players and other encryption-unaware players from accessing bitstreams that need to be decrypted before they can be decoded. For encryption-aware players, however, the new code points contain information on key management and requirements for decrypting encrypted media. In addition they replicate the original codec identifier and other decoding parameters needed to decode the bitstreams once they have been decrypted.

Encrypted 3GP files can also be used for streaming servers to serve encrypted media over RTP. Hint tracks of such 3GP files are not encrypted per se, i.e. a PSS server does not have to decrypt anything in order to serve the encrypted content. Information on key management and decryption is conveyed to the client in the SDP description, with the relevant parts stored in the hint track of the 3GP file. However, as the content provider may want to force the server to take certain actions, such as providing integrity protection before data is streamed, there is still a need to redefine the code point for hint tracks as well. The new code points replicate the original code point information while providing information on required integrity protection. This way encryption-unaware servers will be prevented to serve encrypted data that were supposed to be integrity protected.

3.1 Profile for encrypted 3GP files

The Encryption profile (branded ‘3ge6’) is defined for 3GP files that contain encrypted media. Further details on the kind of file that is encrypted is given by other brands, such as a Basic profile brand for download of audio/video presentations or Streaming-server profile for serving of encrypted content.

Files conforming to Encryption profile shall use the encrypted-sample description entries (code points) for media tracks containing encrypted media. A file conforming to Encryption profile may contain both encrypted and unencrypted tracks.

The Encryption profile should be used as a major brand. It can also be used in combination with other 3GP profiles, as long as the file conforms to those profiles. In particular,

- Encryption and Basic profiles together imply that the maximum number of tracks shall be one for video, one for audio and one for text. A file may contain both encrypted and unencrypted tracks (but not if they are of the same media type). Note however, that an encryption-unaware player will ignore encrypted tracks.

- Encryption and Progressive download profiles together imply that the file is both encrypted and suitable for progressive download.

- Encryption and Streaming-server profile imply that the content referred to by one or more hint tracks is encrypted. If a PSS server is required to take special actions, such as provide integrity protection, then encrypted sample description entries (code points) for hint tracks shall be used.

Note that the General profile is defined as a superset of all profiles including Encryption profile. A 3GP file conforming to General profile (only) may contain any number of encrypted tracks not yet combined into 3GP files suitable for download or streaming or without necessary information on key management.

The Encrypted-basic profile is a 3GP profile and should be used with the file extension ‘.3gp’.

3.2 Code points for encrypted media

The sample description entries of a media track in a 3GP file identify the format of the encoded media, i.e. codec and other coding parameters. Hence, by simply parsing the sample descriptions, a player can decide which tracks it is able to play.

All sample entries for audio and video derived from the ISO base media file format contain a set of mandatory fields. In addition, they may contain boxes specific to the codec in question. MPEG-4 codecs (Visual and AAC) use the ESDBox, whereas AMR and H.263 use the AMRSpecificBox and the H263SpecificBox, respectively.

The principle behind storing encrypted media in a track is to “disguise” the original sample description entry with a generic code point for encrypted media. We define three code points (four-character codes of the sample description entries) for signalling encrypted video, audio and text as follows:

format identifier	original format	media content
encv	s263, mp4v	encrypted video: H.263 or MPEG-4 visual
enca	samr, sawb, mp4a	encrypted audio: AMR, AMR-WB or AAC
enct	tx3g	encrypted text: timed text

The “encrypted” versions of the sample descriptions replicate the original sample descriptions and include a protection information box with details on the original format as well as all requirements for decrypting the encoded media. The EncryptedVideoSampleEntry and the EncryptedAudioSampleEntry are defined in Tables 3.1 and 3.2, where TheProtectionInfo box is simply added to the list of boxes contained in a sample entry.

Table 3.1: EncryptedVideoSampleEntry

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		‘encv’
All fields and boxes of a visual sample entry, e.g. MP4VisualSampleEntry or H263SampleEntry.			
ProtectionInfoBox		Box with information on the original format and encryption	

Table 3.2: EncryptedAudioSampleEntry

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		‘enca’
All fields and boxes in an audio sample entry, e.g. MP4AudioSampleEntry or AMRSampleEntry.			
ProtectionInfoBox		Box with information on the original format and encryption	

The EncryptedVideoSampleEntry and the EncryptedAudioSampleEntry can also be used with any additional codecs added to the 3GP file format, as long as their sample entries are based on the SampleEntry of the ISO base media file format.

The EncryptedTextSampleEntry is defined in Table 3.3. Text tracks are specific to 3GP files and defined by the Timed text format in 26.245. In analogy with the cases for audio and video, we add a ProtectionInfoBox at the end.

Table 3.3: EncryptedTextSampleEntry

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		‘enct’
All fields and boxes of TextSampleEntry.			
ProtectionInfoBox		Box with information on the original format and encryption	

3.3 Key management

The necessary requirements for decrypting media is stored in the Protection information box. It contains the Original format box, which identifies the codec of the decrypted media, the Scheme type box, which identifies the protection scheme used to protect the media, and the Scheme information box, which contains scheme-specific data (defined for each scheme). The Protection information box and its contained boxes are defined in Tables 3.4 – 3.7.

Table 3.4: ProtectionInfoBox

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		'sinf'
BoxHeader.Version	Unsigned int(8)		0
BoxHeader.Flags	Bit(24)		0
OriginalFormatBox		Box containing identifying the original format	
SchemeTypeBox		Box containing the protection scheme.	
SchemeInformationBox		Box containing the scheme information.	

Table 3.5: OriginalFormatBox

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		'frma'
DataFormat	Unsigned int(32)	orginal format	

DataFormat identifies the format (codec) of the decrypted, encoded data. The currently defined formats in 3GP files include 'mp4v', 'h263', 'mp4a', 'samr', 'sawb' and 'tx3g'.

Table 3.6: SchemeTypeBox

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		'schm'
BoxHeader.Version	Unsigned int(8)		0
BoxHeader.Flags	Bit(24)		0 or 1
SchemeType	Unsigned int(32)	4cc identifying the scheme	
SchemeVersion	Unsigned int(16)	Version number	
SchemeURI	Unsigned int(8)[ ]	Browser URI (null-terminated UTF-8 string). Present if (Flags & 1) true	

SchemeType and SchemeVersion identifiy the encryption scheme and its version. An example that can be used for OMA DRM is given in the following section. As an option, it is possible to include an URI pointing to a web page for users that don't have the encryption scheme installed.

Table 3.7: SchemeInformationBox

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		‘schi’
BoxHeader.Version	Unsigned int(8)		0
BoxHeader.Flags	Bit(24)		0
		Box(es) specific to scheme identified by SchemeType	

The boxes contained the SchemeInformationBox are defined by the scheme type.

3.4 Example encryption scheme

The encryption scheme to be used in conjunction with OMA DRM needs to be defined. In section **Error! Reference source not found.** we propose the use of AES_CM_ES. OMA should provide input on the file format boxes expressing the scheme in 3GP files, specifically on the requird additional headers. Below is an example of how such a definition may look like:

- Scheme type: ‘odkm’
- Scheme version: 0x0200
- Scheme-specific boxes: OMADRMSampleFormatBox and OMADRMCommonHeadersBox, see Tables 3.8 – 3.9.

Table 3.8: OMADRMSampleFormatBox

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		‘osfm’
BoxHeader.Version	Unsigned int(8)		0
BoxHeader.Flags	Bit(24)		0
SelectiveEncryption	Bit(1)		0 or 1
Reserved	Bit(7)		0
KeyIndicatorLength	Unsigned int(8)	Length of key indicator	
IVLength	Unsigned int(8)	Length of IV	

Table 3.9: OMADRMCommonHeadersBox

Field	Type	Details	Value
BoxHeader.Size	Unsigned int(32)		
BoxHeader.Type	Unsigned int(32)		‘odhe’
BoxHeader.Version	Unsigned int(8)		0
BoxHeader.Flags	Bit(24)		0
EncryptionMethod	Unsigned int(16)	Encryption method	
EncryptionPadding	Unsigned int(16)	Padding type	
PlaintextLength	Unsigned int(32)	Plaintext content length in bytes	
ContentIDLength	Unsigned int(16)	Length of ContentIP field in bytes	
RightsIssuerURLLength	Unsigned int(16)	Rights Issuer URL field length in bytes	
TextualHeadersLength	Unsigned int(16)	Length of the TextualHeaders array in bytes	
ContentID	Unsigned int(8) [ContentIDLength]	Content ID string	
RightsIssuerURL	Unsigned int(8) [RightsIssuerURLLength]	Rights Issuer URL string	
TextualHeaders	Unsigned int(8) [TextualHeadersLength]	Additional headers as Name: Vaule pairs	
ExtendedHeaders		Extensible headers to end of box (future use)	

3.5 Encrypted server files

PSS servers can also use 3GP files for streaming of encrypted media. The principle here is to packetise-then-encrypt. Conceptually, there is no difference between serving encrypted media and unencrypted media from a 3GP server file. In both cases, the PSS server can simply follow the hint instructions of the file. All the necessary information for using the streamed media is conveyed to the client via the SDP description. For encrypted media this also includes the requirements for decrypting the media streams.

4 References

[1] 3GPP TS 22.242 v2.0.0 (DRM Stage 1 document), ftp://ftp.3gpp.org/tsg_sa/WG1_Serv/TSGS1_16_Victoria/Output/S1-021185.zip

[2] LS on Digital Rights Management (from 3GPP SA to OMA), September 2002, see http://www.3gpp.org/ftp/tsg_sa/TSG_SA/TSGS_17/Docs/PDF/SP-020626.pdf

[3] OMA-DLDRM-2003-0081R2-3GPP-SA4-liaison-01May2003, “Liaison on DRM content format from OMA DLDRM to 3GPP SA4”

[4] OMA-MAG-DLDRM-2003-0172R1-liaison-to-3GPP-SA4, ” Liaison to 3GPP SA4”

[5] OMA-BAC-DLDRM-2003-0221R3-liaison-to-3GPP-SA4-and-SA3, “Liaison to 3GPP SA4 and SA3 on issues on DRM for PSS and MBMS streams”

[6] TS 26.244

3GPP TSG-SA WG3 Meeting #xx
xxx, xxx, xxx yyy 2003

Tdoc S3 (03) 0xxx
Agenda Item: x.x.x

Source: Ericsson
Title: Real-time transport of protected continuous PSS media
Document for: Discussion

1 Introduction

3GPP has delegated the standardization of DRM [1] to OMA [2]. However, it turned out that for interoperability of 3GPP PSS streaming and 3GPP MBMS with OMA DRM 2.0, adaptations on both ends are necessary. OMA has proposed that 3GPP defines the protected file format and the streaming mechanisms for protected PSS media [3][4], and key management is handled in the framework of the OMA DRM 2.0 specification. According to the requirements laid out in [3], media tracks are encrypted and stored in a 3GP file. The 3GP file can be downloaded as a whole, or encrypted packets can be extracted from the 3GP file and transported to the client using real-time transport protocols and mechanisms (that means transport protocols based on RTP/UDP).

This input proposes methods for real-time streaming of protected media with confidentiality and integrity protection. Changes to the 3GP file format [16] that allow the storage and download of protected / DRM encrypted PSS media are not considered here, but are subject of a separate input (Ericsson: Extensions to the 3GP file format for storage of encrypted / DRM protected media).

Since the media streams / tracks / packets are encrypted, they are not any longer compliant to the RTP payload formats defined by the IETF and used in 3GPP PSS [6][7][8]. Thus, it is necessary to define a mechanism that can transport encrypted payloads, specifically encrypted versions of [6][7][8], but preferably also any other defined RTP payload format. In general, encryption of data without in detail analysing the security setting does not necessarily give confidentiality. There are many other mistakes that can made, in particular when optimisations are attempted, e.g. to support a capability limited mobile streaming client.

To start from scratch and specify security for streaming would require a considerable investigation and is not just a matter of specifying a crypto suite. Key derivation, implications of including or omitting integrity protection, protection of RTP headers, replay protection and protection against man-in-the-middle attacks are just examples of considerations that have to be made. Thus, we recommend that the solution 3GPP adopts relies as much as possible on scrutinized security mechanisms and protocols. If no perfectly suited solutions exist, small and well-understood amendments to scrutinized standards seems reasonable. This will reduce the effort needed for a security study, although the changes made must be analysed.

This document makes a proposal for real-time streaming of protected PSS media. It extends the secure real-time transport protocol (SRTP), which has undergone an in-depth security review in IETF. This proposal allows to stream PSS media in a way that inter-operates with OMA DRM, and especially with the key management of OMA DRM. This fits to model that the OMA DLDRM group has outlined in their DRM 2.0 specification under development: key management is handled in the framework of the OMA DRM 2.0 specification, while stream protection, stream storage, stream transport and PSS related signalling are handled in the framework of the PSS Rel6 specification.

2 Overview

2.1 Basic Idea

The basic idea is to use a modified version of SRTP for encryption and integrity protection. The modification allows pre-encryption in the content provider trusted zone and decryption in the client trusted zone, while integrity protection can be terminated outside the trusted zone. By deriving encryption and integrity keys from a master key, only one key needs to be conveyed to the consuming client.

There already exists a detailed study on security for streaming done in the IETF audio video transport (AVT) working group: The Secure Real-Time Transport Protocol (SRTP). SRTP is about to become RFC and is currently with the IESG for approval.

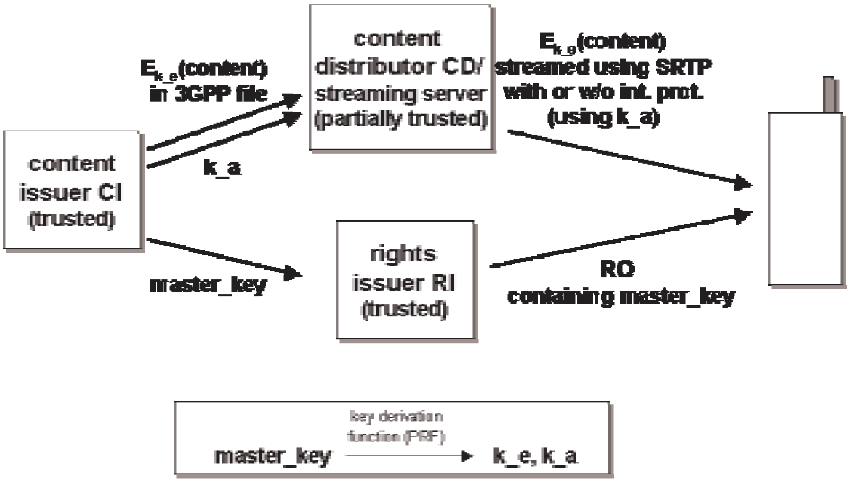
A previous concern about the appropriateness of SRTP for security of DRM media was that the pre-defined transforms in SRTP do not support pre-encryption. However, our proposal overcomes this limitation.

SRTP is a streaming security framework in the sense that it supports the extension of new cryptographic transforms. We propose a pre-encryption transform for SRTP that allows pre-encryption of content at the content provider site and before content is delivered to the streaming server. The transform is also designed with the DRM trust model in mind and allows a definition of restricted trust zones both on the sending and receiving sides. On the sending side this allows streaming servers distributing pre-encrypted content to be located outside the fully trusted domain of the content provider. On the receiving side this allows for a flexibility in the streaming client implementation to accommodate the DRM trust model.

The main advantages with this proposal is the reuse of the existing SRTP specification,which provides a security extension of RTP, designed with wireless and limited processing capacity in mind and where an extensive security analysis has been made and is documented. Only small extensions in the form of simple key material processing (and the addition of the pre-encryption transform of course) are needed in addition to an existing SRTP implementation. The proposal employs packetization prior to encryption.

This input assumes that a method for storage of encrypted / protected PSS media in 3GP file formats is available, for example following a separate proposal from Ericsson (Extensions to the 3GP file format for storage of encrypted / DRM protected media).

Figure 1 shows the basic idea and how this input related to the file format proposal (Ericsson: Extensions to the 3GP file format for storage of encrypted / DRM protected media). The content provider uses a master key master_key. From the master key the integrity key k_a and content encryption key k_e (CEK in OMA terminology) can be derived using a key derivation function. The encryption is done at the content provider. The streaming server receives the encrypted content in a 3GPP file and the integrity key k_a (if the CP/RI choose to apply integrity protection). The streaming server then streams the content to the client using SRTP. No additional encryption is applied. If applicable, the streaming server applies integrity protection. The content provider conveys the master key master_key to the RI. The RI issues a RO to the client, which contains the master key master_key. From master_key and knowing the key derivation function, the client can derive the content key k_e and (if applicable) the integrity key k_a.



2.2 Motivation for the use of SRTP for Streaming DRM

We believe the use of SRTP as a basis for our DRM security proposal for PSS streaming has advantages, which make it favourable over the use of an encrypted RTP container format combined with a separate integrity protection mechanism (possibly SRTP). We think our proposal has the following distinct advantages:

- Encryption and integrity protection are achieved using components from the same mechanism (SRTP), thus there is no need to separately implement confidentiality and integrity protection mechanisms
- The computational complexity is comparable to competing proposals with separated encryption and integrity protection mechanisms
- SRTP is a scrutinized and open proposed RFC (it is expected to shortly become RFC in IETF). It seems advantageous to base the DRM solution on a solid and future-proof standard/RFC.
- Only one key needs to be conveyed in the RO, and no other second key conveyed out of band, as would be otherwise necessary. In our proposal, one key is conveyed from which encryption key and integrity key are derived.
- SRTP allows to transport any defined RTP payload format since the SAVP profile indicates encrypted payload.
- An open-source SRTP implementation is available under a BSD-based license [14]

.

2.3 The Roles

This DRM for streaming solution contains a number of different roles and entities in the chain of processing.

- Content Issuer (CI) – Encodes and packetizes the content. To protect the content the CI does pre-encryption of the packetized content.

- The content distributor (CD), i.e. a streaming server, streams the pre-encrypted content and optionally applies integrity protection. The streaming server may be within the trusted boundary of the CI or in a domain with lower trust, that means not trusted to keep the confidentiality of the content (see restricted trust zones).

- Rights Issuer (RI) – has close trust relation to CI and is authorised to issue Rights Objects (ROs) to DRM compliant clients

- Streaming client/DRM agent – requests/receives protected media and ROs, checks possible integrity protection and decrypts the streaming media.

2.4 Restricted Trust Zones

The assumed trust zones for normal SRTP (protection of conversational media) and DRM are different:

- In SRTP and conversational scenarios the confidentiality and integrity protection is only needed between the two end-points of the communication. Thus the application space on either end is trusted.

- In a DRM protected distribution the sender of the RTP packets may not be trusted by the content issuer. Thus content confidentiality for the content distributor is needed to be available.

- For DRM enabled consumer of content the receiving and displaying application is not fully trusted. To minimize the risk for leakage of confidential information, either media or keys, the part of the application required to be trusted is to be minimized.

Thus we have three types of trusts in the solution:

- Fully Trusted: This trust relation allows access to all types of keys, unprotected media. Examples of these parts are, the content issuer where he creates, packetize and protects the media, and the Content consumers DRM agent and media decryption, decoding and displaying facilities.

- Partially trusted: This trust relation does not allow access to the protected content, however the trust is given to ensure that media is delivered in the correct way. The entities given this trust are assumed to not trying to hurt the content processed. Examples of parts given this trust is the content distribution (streaming server) and the rest of the receiving application.

- Untrusted: No trust at all are placed in these relation. Example of this is the complete network between content distributor and content consumer.

In some cases the trust relations may be simpler, for example a streaming server may be fully trusted, thus allowing unencrypted media to be stored on the server for complete DRM protection processing with the streaming server instead of divided between the CI and the CD.

2.5 Scenario Walkthrough

This section outlines the flow of the content and keys through the different roles and stages. Confidentiality protection is added as an additional layer between the RTP stack and the packetization layer. The exact behaviour is specified in Section 5.

The setup phase:

> The CI packetizes media, encrypts packetized media and puts in hint tracks. When this is done, the CI forwards the media to the streaming server (CD). If authentication is going to be used, additional authentication information (integrity key) is forwarded to the CD.

The RI obtains information of media, protection keying material (master_key) and usage rights from CI and prepares licenses (OMA Rights Objects).

The Content distribution phase:

1. The Client requests media from CD through RTSP and receives the SDP. The SDP contains information necessary to run SRTP, the DRM key management information (including link to RI) and other necessary media setup information.

2. The Client request to buy rights from RI. The RI checks the Client and if compliant issues a RO to it.

3. The Client sets up a streaming session with CD using RTSP. Information about destination address for RTP session and SSRC to be used by the CD is agreed on.

4. The CD starts sending RTP packets from the hint-track. If integrity protection is used, SRTP protection is applied using the keying material received from the CI. No encryption (i.e., the pre-defined NULL-encryption algorithm) is applied by SRTP in CD.

The Content Reception phase:

1. The Client receives the encrypted (and possibly integrity protected) packet.

2. The SRTP stack performs its reception processing, i.e., perform NULL-decryption and check and remove integrity protection, etc using keying material received from the trusted zone.

3. Perform normal RTP processing. Including removal of padding if needed.

4. Decrypt the packets payload (according to permissions granted in the usage rights conveyed in the RO) in the trusted zone using the inverse of the pre-encryption transform and forward the unencrypted payload for depacketization and consumption.

3 Media encryption and real-time media transport / Proposed SRTP solution

3.1 Introduction

The Secure Real Time Transport Protocol [10]is a profile of the Real Time Transport Protocol (RTP), which can provide confidentiality, message authentication, and replay protection to the RTP/RTCP traffic.

SRTP is a framework, which permits upgrading with new cryptographic transforms. Section 6 of [10]provides guidelines to add a transform to SRTP, through a companion specification.

This section outlines a proposed new transform of SRTP, which supports pre-encryption of packetized streaming media. This allow for content confidentiality between both, CI to streaming server distribution, and for transmission as RTP packets between streaming server and client.

A new step in the processing is added, which performs the pre-encryption of the media. This is normally performed by the content issuer, rather than the content distributor.

Assuming that the streaming server (CD) has an existing SRTP implementation, this solution does not change the sending side, except for introducing a conceptual “NULL” key derivation (since the key is used directly for authentication, not going through the usual SRTP key derivation). The encryption transform used between the SRTP-stacks on CD and the client is the pre-defined NULL encryption with optional integrity protection of the RTP packets. Note that integrity protection of the RTCP stream is mandatory. On both the sending and the receiving side the decryption transform for SRTP is set to the NULL-transform. The real decryption is added in an additional processing step that is performed after RTP processing and before (de)packetization. This decryption utilize the bulk of the already in SRTP defined AES_CM, but with a new explicit packet counter to derive the Initialisation Vector (IV).

This results in a receiver side with a possible SRTP stack implementation according to Figure 1. The SRTP performing the integrity protection can have all the capabilities according to SRTP, and can thus also be used for other purposes if needed. The consideration in implementation for our specific purposes is that the SRTP key-derivation and key context containing the master key must be in the trusted zone together with the decryption process.

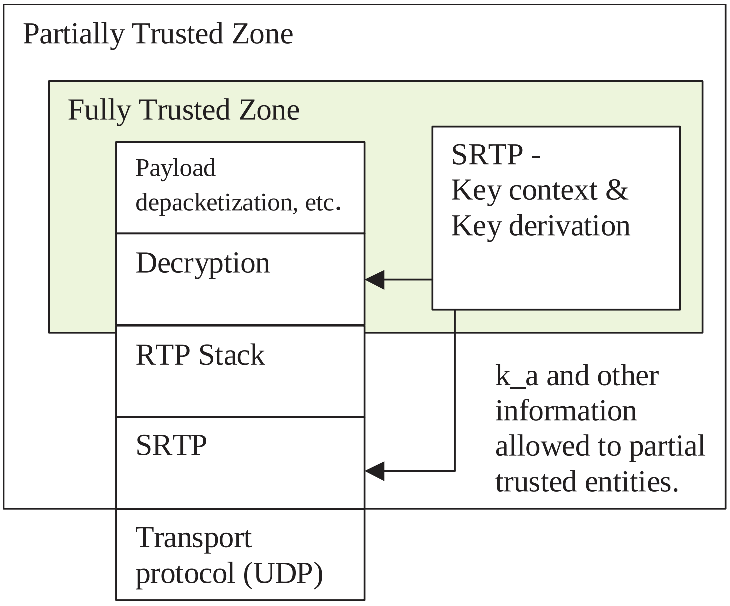

Partially Trusted Zone
Fully Trusted Zone
Payload depacketization, etc.
Decryption
RTP Stack
SRTP
Transport protocol (UDP)
SRTP - Key context & Key derivation
k_a and other information allowed to partial trusted entities.

Figure 1 - Stack view for proposed solution

3.2 RTP packet structure

The RTP packet as seen when transported over the network for this transform is identical to the SRTP packet. However, note that the payload in reality consists of two distinct parts:

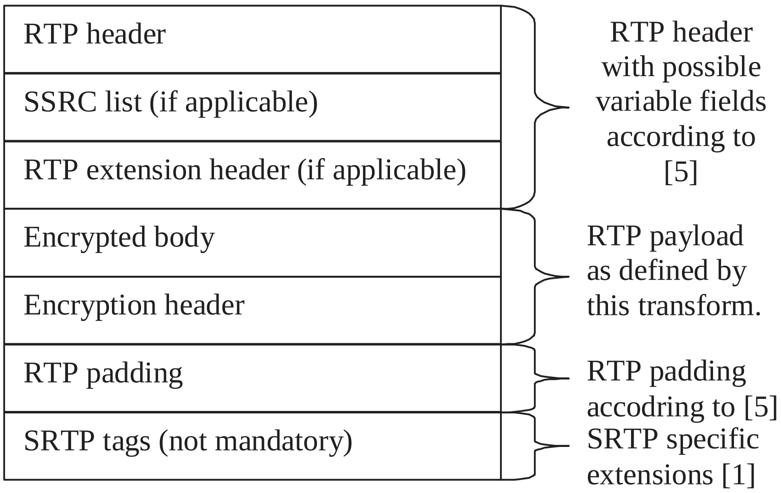

RTP header
SSRC list (if applicable)
RTP extension header (if applicable)
Encrypted body
Encryption header
RTP padding
SRTP tags (not mandatory)
RTP header with possible variable fields according to [5]
RTP payload as defined by this transform.
RTP padding accodring to [5]
SRTP specific extensions [1]

Figure 2 - SRTP packet with headers, payload and profile extension

The RTP header with possible variable length fields and extension headers, or profile specific definitions comes first in the RTP packet. Followed by this RTP payload formats two parts as indicated with the encrypted body first, followed by the encryption header. The SRTP profile allows two non-mandatory fields after the payload: The Master Key Index (MKI) for use in key management and an Authentication Tag for integrity protection.

3.2.1 Payload structure

The payload in the proposed SRTP transform consists of the two parts:

- The Encrypted body
- The Encryption header

The encrypted body SHALL precede the encryption header in the RTP payload part.

The Encrypted body SHALL consist of the encrypted bits of any RTP payload format, including payloads such as redundancy format [4]. The encryption algorithm SHALL be AES in Segmented Integer Counter Mode (AES-CM) with 128 bits key, with IV as defined below.

The Encryption header SHALL consist of a Packet counter (PC) of size 32 bits used in the encryption algorithm (IV).

The same core encryption algorithm AES-CM is used in the pre-defined SRTP transform but there are two main differences:

- The payload is pre-encrypted by the CI, so no encryption is done on the fly at the server (see Packet processing below).
- The counter used in the encryption algorithm is included in the Encryption header (see Encryption below) as opposed to the predefined transform where it depends on the RTP header in a way that prevents pre-encryption. This affects IV formation, but this is anyway part of the transform specification.

Another difference compared to the pre-defined SRTP transform is that the size of the payload of the SRTP packet is larger than the corresponding RTP payload due to the Encryption header (the size of the Encrypted body is the same as the cleartext RTP payload). Note that having a larger ciphertext than plaintext is a property of other block cipher modes, such as CBC, as well. Care should be taken to avoid exceeding the MTU size when using this transform.

3.2.2 RTP packet in detail

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
  |V=2|P|X|  CC   |M|     PT      |       sequence number         | |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
  |                           timestamp                           | |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
  |           synchronization source (SSRC) identifier            | |
  +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ |
  |            contributing source (CSRC) identifiers             | |
  |                               ....                            | |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
  |                   RTP extension (OPTIONAL)                    | |
+>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
| |                          payload  ...                         | |
| |                                                               | |
| |                                                               | |
| |                               +-------------------------------+ |
| |                               |       Packet Counter (PC)     | |
+>|-------------------------------+-------------------------------+ |
| |    Packet Counter (PC)        | RTP padding   | RTP pad count | |
| +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<+
| ~                     SRTP MKI (OPTIONAL)                       ~ |
| +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
| :                 authentication tag (RECOMMENDED)              : |
| +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                                                                   |
+- Encrypted Portion*                      Authenticated Portion ---+
```

For the RTP stack the Packet Counter seem to be part of the RTP payload and is not changed by the RTP stack. There should normally not exist any reasons for using RTP padding. However with this solution it is possible to use it as normal and let the RTP stack handle the addition and removal of the padding. SRTP might on the sender side need to adjust the padding to meet the block sizes allowed when adding any SRTP tags to the packet.

The use of general FEC [3] is also possible however the processing point must be defined since SRTP / FEC co-existence can be configured. It can be in either of three places:

1. Add FEC to already encrypted and integrity protected payloads. Will result in that FEC must be resolved prior to Integrity checks on the receiving side. This allows for e.g. a DoS attack to use larger mounts of receiver resources, and also results in that the FEC packets become unnecessary large.

2. Add FEC after encryption but prior to integrity protection. Prevents the receiver to waste resources on FEC recovery operation for non-protected packets. Result in the recovery of complete RTP packets that need further processing. Does not add redundancy to cleartext prior to encryption, which is good cryptographic practice.

3. First add FEC prior to encryption and integrity protection. Does not work with FEC as defined in RFC 2733 due to that the RTP TS is not given at the time of encrypting the payload, thus the recovery operation can't be performed correctly.

Therefore it would be strongly recommended that FEC operations are performed according to alternative 2.

3.3 Packet processing

3.3.1 Pre-Encryption processing

The pre-encryption step packet processing SHALL be done in the following way. Input to the processing is full formed RTP payloads, packetized according to the media format's specification.

The encryption of the payload SHALL then be performed using the derived (see Section 5.6) encryption key (k_e), a session salt, k_s, a unique packet counter for each payload, and the encryption algorithm as specified by Section 5.4. The output from this encryption step is then taken and at the end the unique packet counter used to encrypt is added. This forms the new payload.

3.3.2 Sender Side

The SRTP processing on the sender side assumes that the RTP payload being sent through the RTP stack down to the standard SRTP stack is already encrypted according to section 5.3.1.

Thus, the packet processing SHALL be the same as defined in Section 3.2 and 3.3 of [10], for SRTP and SRTCP respectively, using NULL encryption and optionally the integrity protection scheme defined in section 5.5.

3.3.3 Receiver Side

The packet processing SHALL be the same as defined in Section 3.2 and 3.3 of [10], for SRTP and SRTCP respectively, but note the following :

- When performing step 4 (Decrypt payload) of the reception process in section 3.3, the NULL-transform MUST be applied. In other words, no “real” decryption takes place at this stage.

Note that the payload of the RTP-packet that is the result of the above SRTP processing is still in encrypted form. The RTP-packet is then processed by the normal RTP stack, and the resulting payload is passed upwards. We are now left with the encrypted payload, which carries the PC as a trailer. The encrypted payload and PC are fed to an additional “decryption layer”, which performs the actual decryption of the media payload as specified in Section 5.4. When the media is decrypted, the PC is removed and it is passed to the codec.

As described above, one way to view the solution is that the decryption algorithm used is actually the NULL-transform, and that a new decryption layer that mimics the SRTP decryption process is inserted above the RTP layer. An alternative view is that the (real) decryption stage is moved from the SRTP layer to above the RTP layer. That is to say, the SRTP implementation is now on both sides of the RTP implementation. No matter which point of view is taken, the effects on a standard SRTP implementation is the same.

Note: If the streaming server (CD) is located in the fully trusted zone (e.g. CI=CD) then it can use SRTP with predefined default transform AES_CM and encrypt on-the-fly. If the SRTP stack in the client is located within the fully trusted zone then the pre-encryption transform as well as the (entire) key derivation MAY be co-located with SRTP, there replacing the NULL transforms.

3.4 Contexts

We now describe how to handle SRTP cryptographic contexts such that an existing SRTP implementation below RTP in the stack can be totally re-used on both sending and receiving side. As noted below, there may be other approaches to actual implementation, though they will be input-output compatible with the following description.

Conceptually the CI and the trusted zone of the client have a “primary crypto context”, which contains all information necessary to encrypt and authenticate the media. This context is compatible with a standard SRTP context and includes, e.g. the master key, master salt and the pre-defined PRF as defined in [8]. From a primary context, a special reduced SRTP context can be derived. The reduced SRTP context will have the master salt, the PRF set to the identity mapping, and master key of the context set equal to the authentication key derived from the master key. Such an SRTP context can be pushed down from the trusted zone to the SRTP implementation in the partially trusted zone. The reduced SRTP context is still a full SRTP context in accordance with [10], but is a projection of the primary context, i.e. the information is reduced to the bare minimum needed to perform the authentication. The primary context is exactly the same as for AES_CM defined in [10].

The CI will send the reduced SRTP context that includes the identity mapping PRF and the authentication key as master key to the streaming server. This will allow the SRTP implementation to obtain the correct integrity key, but it cannot access the decryption key.

In the trusted zones (at the decryption layer in the client and at the CI), the primary context is used to perform the confidentiality protection since both encryption and integrity keys can be derived at this level.

Note that this view is only conceptual, and an implementation will typically not be involved with primary and reduced contexts.

3.5 Confidentiality Protection

This Section extends Section 4 of [10]. To allow pre-encryption, a special cipher transform is defined. Note that the encryption is applied at the CI, and not at the CD. Hence the SRTP implementations on the CD and the client both use NULL-encryption, but the new decryption-layer in the client decrypts the actual media.

3.5.1 Cipher

To allow pre-encryption in the SRTP framework, we have added an additional confidentiality layer above RTP. We define a new confidentiality transform according to SRTP specifications, but we do not actually plug this transform into the SRTP implementation, but rather let it run in the decryption layer in the client and at the CI when performing pre-encryption.

Cipher-id = AES_CM_EC

AES_CM_EC (Explicit Counter) SHALL use AES in Segmented Integer Counter Mode (AES_CM) with 128 bits key and IV as specified below. This transform coincides with the predefined AES_CM in all but one thing, the IV construction.

3.5.2 Keystream generation

The description of usage for AES-CM in Section 4.1.1 of [10] is valid with the exception of the IV, which MUST be replaced by

IV = (k_s * 2^16) XOR (PC * 2^16),

where PC is the Packet counter in the Encryption header field.

The reason for this IV definition is that the default IV of [10] depends on SSRC and SRTP packet index i. This would make generation of the IV in advance at the content provider side impossible.

Note that the index of SRTP is 48 bits long (the 16-bit SEQ field from the RTP-header concatenated with the 32-bit rollover counter), implying that 2^{48} packets can be encrypted before the key needs to be changed. Since the PC (which has the same purpose as the index in the pre-defined transforms) is only 32 bits long, “only” 2^{32} packets can be encrypted with   AES_CM_EC before the key needs to be changed.

3.6 Data integrity and replay protection

When applied, this is done exactly as in SRTP using the standard SRTP transforms on both server and client side, , but as noted, with the exception of the key derivation. Since the session integrity key is pushed into the SRTP implementation directly, both server and client need to run a special PRF (see Section 3.7.1), which is the identity mapping.

Note that the SRTP ROC (roll-over counter) is included in the authentication coverage (as defined in SRTP) and so is the packet counter, PC. Since the ROC (which is part of the packet index) is included in the authentication coverage, robust replay protection can be provided as specified by SRTP.

The integrity transform (when applied) SHALL be HMAC with SHA-1 and a MAC length of 32 bits.

3.7 Key derivation

3.7.1 PRF

SRTP requires a key derivation function PRF to be defined, see Section 4.3 of [10] and key derivation to be executed.

PRF depends on two variables PRF(k,x) where k is a master key for this SRTP implementation and x depends on master_salt, <label> and other things. <label> is used to indicate what session key should be derived, encryption key (k_e), session salt (k_s) or authentication key (k_a) and if for SRTP or SRTCP. The master keys for SRTP and SRTCP may be different (Section 3.2.3 of [10]). The definition of PRF is a part of the crypto context, we will use this option to redefine PRF for our purposes. There is a default PRF defined in Section 4.3.3 of [10].

Note that interface to the derivation function is fixed though the definition of the function may be altered.

3.7.2 Allowing partially trusted zones

We must cope with the scenario that there are different trust levels with respect to encryption/decryption and integrity protection, i.e. that k_e, and k_s used to encrypt the content have restricted to the fully trusted zone whereas k_a is available also in the partially trusted zone.

Depending on implementation of a DRM scenario, there may be SRTP implementations in the partially trusted zone that are not trusted with the master key, k_e or k_s (doing integrity protection but not encryption/decryption) but requires a PRF and a “master key” to perform the key derivation.

For this purpose we consider the following construction:

1. The true master key is available only in the fully trusted zone

2. Using the SRTP default PRF, generate k_e, k_s and k_a.

3. For the SRTP implementation in the partially trusted zone, the following trivial key derivation function SHALL be used:

   PRF’(k,x) = k

   This is well-defined for all <label> values (see [10]), but in practice only <label>=0x01 and <label>=0x04 will be used, these labels are used to derive authentication keys).

4. By defining this key derivation function PRF’ and providing the authentication key as “master key”: master_k’ = k_a to an SRTP implementation in a partially trusted zone, the implementation will derive the same authentication key as was derived from the true master key.

5. Thereby by just solving key management for the master_key (as previously described using the OMA DRM Rights Objects) both the same encryption and the integrity keys are available on the sending and receiving sides.

Note that one might consider using the abbreviation “PRF” (Pseudo random function) when referring to an identity mapping as “abuse of notation”, but this notational convention/simplification is convenient in this case.

The default PRF SHALL be used in the primary context and PRF’ SHALL be used in the reduced SRTP context (see Section 5.4).

3.7.3 Sending side

This section describes proposed key management in a partially trusted zone scenario. Compare the Scenario Walkthrough section.

The setup phase:

1. The CI generates a random master_key and master_salt and derives the session keys k_e, k_s and k_a using the key derivation function PRF according to Section 4.3 of [10]. The packetized media is encrypted using k_e and k_s.

2. The CI forwards media and the “master key” master_key’ = k_a to the streaming server (CD). Although not needed, it is also given the master salt since the client requires the salt, but cannot obtain it via the RO. The key derivation function for the CD SRTP implementation is the function PRF’ defined in the previous section. In

other words the CI send the reduced SRTP context to the CD, in Section 5.4 terminology.

3. The RI obtains master_key from CI and prepares the RO, OMA DRM Rights Objects, (with CEK or REK = master_key).

The Content distribution phase:

1. The Client requests media from CD and receives the SDP containing the master_salt, and the SSRC in the RTSP SETUP response to be used by CD.

2. The Client request to buy rights from RI. The RI checks the Client and if compliant issues a RO (including master_key) to it, protected with the Client public key, i.e., the primary context is inserted in the trusted zone in the Client.

3. The Client sets up a streaming session with CD.

4. The CD starts sending RTP packets from hint-track. SRTP applies protection using authentication key k_a derived from master_key' using PRF'. No encryption is applied by SRTP in CD.

The Content Reception phase:

1. The Client receives the encrypted and integrity protected packet. Prior to this the Client has received the master_key (from the RO) and the master_salt (in an attribute in the SDP) to the fully trusted zone. The session keys k_e, k_s and k_a are derived using the key derivation function PRF according to Section 4.3 of [10], and the client pushes the reduced SRTP context down to the SRTP implementation in the partially trusted zone.

2. The SRTP stack implementation in the partially trusted zone has received master_key' and master_salt from the fully trusted zone, uses the key derivation function PRF' to derive the authentication key k_a and performs its normal reception processing.

3. Remove the authentication tag if used.

4. Perform normal RTP processing.

5. Decrypt the packets payload in the fully trusted zone using the encryption keys and the packet counter PC in the end of the payload to derive the IV.

6. Remove Packet Counter and forward the unencrypted payload for depacketizing.

3.8 Notes on Selective Encryption

Selective encryption is not included in this proposal, since it is known that selective encryption may introduce security vulnerabilities and this needs further analysis. It is for example known that selective encryption often enables reconstruction of media at very low rendering quality. This could imply a serious threat to users' privacy as it is possible to determine what media they consume. Also, selective encryption without integrity protection enables unnoticed manipulation and re-ordering of packets. For more details, see the input (Ericsson: Considerations on selective encryption and integrity protection for DRM protected PSS media streams).

This proposal can be adapted to support selective encryption, e.g. by including a bit in the payload indicating whether this packet is encrypted or not, although we do not recommend to use selective encryption. Should it be specified however, integrity protection of the streams (payload and packet headers) and integrity protection of the information whether a stream uses selective encryption (which may be contained in the SDP signalling) should be applied.

4 Security considerations

Key replacement MUST occur no later than after 2^32 packets.

Even though SRTP has had lots of scrutiny, we have made some rearrangements amongst the building blocks. We note the following:

- The authentication is unaffected by the rearrangements.
- SRTP only encrypts the payload. In this proposal the payload is encrypted prior to SRTP processing. The only difference to AES_CM is that we use an explicit counter (which is equivalent to the index of SRTP, only 16 bits shorter). This counter is covered by the integrity protection, and therefore confidentiality protection is obtained by a primitive which is input-output compatible with SRTP for a given IV value.
- While a "NULL" key derivation is conceptually performed, the key to which this derivation is applied has gone through exactly the same key derivation as the default in SRTP, albeit performed in the trusted zone.

5 References

[1] 3GPP TS 22.242 v2.0.0 (DRM Stage 1 document), ftp://ftp.3gpp.org/tsg_sa/WG1_Serv/TSGS1_16_Victoria/Output/S1-021185.zip
[2] LS on Digital Rights Management (from 3GPP SA to OMA), September 2002, see http://www.3gpp.org/ftp/tsg_sa/TSG_SA/TSGS_17/Docs/PDF/SP-020626.pdf
[3] OMA-DLDRM-2003-0081R2-3GPP-SA4-liaison-01May2003, "Liaison on DRM content format from OMA DLDRM to 3GPP SA4"
[4] OMA-MAG-DLDRM-2003-0172R1-liaison-to-3GPP-SA4, " Liaison to 3GPP SA4"
[5] OMA-BAC-DLDRM-2003-0221R3-liaison-to-3GPP-SA4-and-SA3, "Liaison to 3GPP SA4 and SA3 on issues on DRM for PSS and MBMS streams"
[6] RFC 2429, "RTP Payload Format for the 1998 Version of ITU-T Rec. H.263 Video (H.263+)"
[7] RFC 3016, "RTP Payload Format for MPEG-4 Audio/Visual Streams"
[8] RFC 3267, "Real-Time Transport Protocol (RTP) Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs"
[9] draft-ietf-avt-rtp-retransmission-08.txt, "RTP Retransmission Payload Format"
[10] SRTP, http://www.ietf.org/internet-drafts/draft-ietf-avt-srtp-09.txt

[11] RFC 2733, "An RTP Payload Format for Generic Forward Error Correction"
[12] RFC 2198, "RTP Payload for Redundant Audio Data"
[13] RFC 3550, "RTP: A Transport Protocol for Real-Time Applications"
[14] Open-source implementation of SRTP, see http://srtp.sourceforge.net/srtp.html
[15] TS 26.234
[16] TS 26.244

Doc# OMA-BAC-DLDRM-2003-0264-liaison-to-3GPP-SA4-and-SA3
Originated by OMA Download+DRM
12 November 2003

3GPP TSG SA WG3 Security — S3#31 **S3-030756**
18 - 21 November 2003
Munich, Germany

LIAISON STATEMENT

Title: Liaison to 3GPP SA4 and SA3 on issues on DRM for PSS and MBMS streams
To: 3GPP SA4 and 3GPP SA3
Cc: 3GPP2 S4
Source: Download+DRM group of the Open Mobile Alliance

Contact(s): Frank Hartung, Ericsson
+49 2407 575389
Frank.Hartung@ericsson.com

Attachments: n/a

1 Overview

The Open Mobile Alliance Download+DRM group (OMA DLDRM) thanks SA3 and SA4 for the ongoing good cooperation and information exchange.

In order to meet our timelines and accelerate the standardisation of DRM protected streaming, we hereby provide you with our current DRM Content Format (DCF) draft specification. It contains draft sections for the discrete DRM Content Format (DCF), the Packetized DRM Content Format (PDCF), which is an amended version of the 3GP file format, and for the PDCF streaming format. The latter section contains a description of a streaming format for protected PSS media based on an encrypted RTP wrapper payload format.

2 Proposal

OMA DLDRM suggests that SA3 and SA4 consider the attached OMA DCF specification for the development of their specifications, specifically for the specification of the streaming mechanism for protected 3GPP PSS media.

Section 5.4.3 in the attached document outlines a possible solution that meets the security requirements of the OMA DLDRM group. We do however acknowledge that SA3 and SA4 may have to consider additional requirements that are not relevant for us, and may thus deviate from our proposal.

We intend to update the DCF specification according to the 3GPP specification when it is available.

3 Requested Action(s)

We kindly request 3GPP SA4 and SA3 to note the information contained in this LS, and to reply in case there are questions or comments. As soon as 3GPP has standardised a streaming mechanism for protected 3GPP streaming services (PSS, MBMS), we would welcome to receive information and specification text. We would appreciate receiving this information as soon as possible, preferably soon after the November SA4 meeting.

The next known OMA DLDRM meeting dates are:

- A series of OMA DLDRM conference calls, week Dec 8-13 2003
- OMA face to face ~~plenary~~meeting, 1-6 February 2004, Los Angeles (USA)

NO REPRESENTATIONS OR WARRANTIES (WHETHER EXPRESS OR IMPLIED) ARE MADE BY THE OPEN MOBILE ALLIANCE OR ANY OPEN MOBILE ALLIANCE MEMBER OR ITS AFFILIATES REGARDING ANY OF THE IPR'S REPRESENTED ON THE "OMA IPR DECLARATIONS" LIST, INCLUDING, BUT NOT LIMITED TO THE ACCURACY, COMPLETENESS, VALIDITY OR RELEVANCE OF THE INFORMATION OR WHETHER OR NOT SUCH RIGHTS ARE ESSENTIAL OR NON-ESSENTIAL.

THE OPEN MOBILE ALLIANCE IS NOT LIABLE FOR AND HEREBY DISCLAIMS ANY DIRECT, INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR EXEMPLARY DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF DOCUMENTS AND THE INFORMATION CONTAINED IN THE DOCUMENTS.

THIS DOCUMENT IS PROVIDED ON AN "AS IS" "AS AVAILABLE" AND "WITH ALL FAULTS" BASIS.

~~OMA DLDRM conference calls, week Dec 8-13~~

Also, we are holding weekly OMA DLDRM telephone conferences.

4 Conclusion

OMA DLDRM suggests that SA3 and SA4 consider the attached OMA DCF specification for the development of their specifications. We intend to update our DCF specification according to the 3GPP specification when it is available.

With best regards,   OMA Download+DRM group

DRM Content Format

Draft Version 2.0 – 03~~1~~-~~October~~November-2003

Open Mobile Alliance

OMA-DRM-DCF-v2_0-~~20030131~~20031103-D

Contents

Figures

Tables

1. Scope

Open Mobile Alliance (OMA) specifications are the result of continuous work to define industry-wide interoperable mechanisms for developing applications and services that are deployed over wireless communication networks.

The scope of OMA "Digital Rights Management" (DRM) is to enable the distribution and consumption of digital content in a controlled manner. The content is distributed and consumed on authenticated devices per the usage rights expressed by the content owners. OMA DRM work addresses the various technical aspects of this system by providing appropriate specifications for content formats, protocols, and rights expression languages.

A number of DRM specifications have already been defined within the OMA. See [DRM], [DRMCF] and [DRMREL]. These existing specifications are referred to within this document as "release 1".

The scope for this specification is to define the content format for DRM protected encrypted media objects and associated metadata. This specification addresses the specific format mechanisims defined in the Release 2 "*Digital Rights Management*" specification [~~DRMv2~~DRM-v2].

2. References

2.1 Normative References

[CREQ] "Specification of WAP Conformance Requirements", Open Mobile Alliance™, WAP-221-CREQ. http://www.openmobilealliance.org/

[DRM] "Digital Rights Management", Open Mobile Alliance™. OMA-Download-DRM-v1_0-20020905-CDRM v1

[DRMCF] "DRM Content Format", Open Mobile Alliance™, OMA-DRM-DRMCF-v1_0

[DRMREL] "DRM Rights Expression Language". Open Mobile Alliance™. OMA-DRM-DRMREL-v2_0. http://www.openmobilealliance.org/

[DRM-v2] "Digital Rights Management". Open Mobile Alliance™. OMA-DRM-DRM-v2_0. http://www.openmobilealliance.org/

[ISO14496-12] "Information technology — Coding of audio-visual objects – Part 12: ISO Base Media File Format", International Organisation for Standardisation, ISO/IEC 14496-12, 2003

[ISO7498-2] TODO

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels". S. Bradner. March 1997. URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC2234] "Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. November 1997. URL:http://www.ietf.org/rfc/rfc2234.txt

[RFC2392] "Content-ID and Message-ID Uniform Resource Locators". E. Levinson. August 1998. http://www.ietf.org/rfc/rfc2392.txt

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee et al. August 1998, http://www.ietf.org/rfc/rfc2396.txt

[RFC2616] "Hypertext Transfer Protocol -- HTTP/1.1". R. Fielding, et al. June 1999. http://www.ietf.org/rfc/rfc2616.txt .

[RFC2630] "Cryptographic Message Syntax". R. Housley. June 1999. http://www.ietf.org/rfc/rfc2630.txt

[TS26.234] "Transparent end-to-end packet-switched streaming service (PSS); Protocols and Codecs", The Third Generation Partnership Project, TS-26.234

[TS26.244] "Transparent end-to-end pPacket-switched sStreaming sService (PSS); File Format", The Third Generation Partnership Project, TS-26.244

[WSP] "Wireless Session Protocol". WAP Forum™. WAP-230-WSP. http://www.openmobilealliance.org/

~~[CREQ]~~ ~~"Specification of WAP Conformance Requirements", Open Mobile Alliance™, WAP-221-CREQ. http://www.openmobilealliance.org/~~

~~[DRM]~~ ~~"Digital Rights Management", Open Mobile Alliance™. OMA-Download-DRM-v1_0-20020905-CDRM v1~~

~~[DRMCF]~~ ~~"DRM Content Format", Open Mobile Alliance™, OMA-DRM-DRMCF-v1_0~~

~~[DRMREL]~~ ~~"DRM Rights Expression Language". Open Mobile Alliance™. OMA-DRM-DRMREL-v2_0. http://www.openmobilealliance.org/~~

~~[DRM-v2]~~ ~~"Digital Rights Management". Open Mobile Alliance™. OMA-DRM-DRM-v2_0. http://www.openmobilealliance.org/~~

~~[ISO-14496-12]~~ ~~"Information technology — Coding of audio-visual objects – Part 12: ISO Base Media File Format", International Organisation for Standardisation, ISO/IEC 14496-12, 2003~~

~~[ISO7498-2]~~ ~~TODO~~

~~[RFC2119]~~ ~~"Key words for use in RFCs to Indicate Requirement Levels". S. Bradner. March 1997.~~

~~URL:http://www.ietf.org/rfc/rfc2119.txt~~

~~[RFC2234]~~ ~~"Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. November 1997. URL:http://www.ietf.org/rfc/rfc2234.txt~~

~~[RFC2392]~~ ~~"Content-ID and Message-ID Uniform Resource Locators". E. Levinson. August 1998. http://www.ietf.org/rfc/rfc2392.txt~~

~~[RFC2396]~~ ~~"Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee et al. August 1998, http://www.ietf.org/rfc/rfc2396.txt~~

~~[RFC2616]~~ ~~"Hypertext Transfer Protocol -- HTTP/1.1". R. Fielding, et al. June 1999. http://www.ietf.org/rfc/rfc2616.txt .~~

~~[RFC2630]~~ ~~"Cryptographic Message Syntax". R. Housley. June 1999. http://www.ietf.org/rfc/rfc2630.txt~~

~~[TS-26.244]~~ ~~"Transparent end-to-end Packet-switched Streaming Service (PSS); File Format", The Third Generation Partnership Project, TS-26.244~~

~~[WSP]~~ ~~"Wireless Session Protocol". WAP Forum™. WAP-230-WSP. http://www.openmobilealliance.org/~~

2.2 Informative References

[WAPARCH] "WAP Architecture". Open Mobile Alliance™. WAP-210-WAPArch. URL:http//www.wapforum.org/

<<If there are no references of a particular type, state that there are none>>

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Asset Content governed by rights. See DRM content.

Box A binary data structure conforming to elementary data type definitions in [ISO14496-12]

Composite object A content object that contains one or more Media Objects by means of inclusion.

Confidentiality The property that information is not made available or disclosed to unauthorised individuals, entities or processes. (From [ISO-7498-2])

Content One or more Media Objects

Content Issuer The entity making content available to the DRM Agent in a device.

Content Provider An entity that is either a Content Issuer or a Rights Issuer.

Content Retailer An entity that is a Content Issuer and/or a Rights Issuer.

Continuous Media Content which is inherently time-based, i.e. might have an implicit or explicit duration and requires multiple iterations of an algorithm to produce a continuous media experience to a User, such as video or audio.

Device A Device is a user equipment with a DRM Agent. The Device MAY include a smartcard module (e.g. a SIM) or not depending upon implementation.

Discrete Media Content that can be rendered with a single pass of an algorithm to interpret the media content, media that itself does not contain an element of time, such as still images or web pages

DRM Agent The entity in the Device that manages Permissions for Media Objects on the Device.

DRM Content Content that is consumed according to a set of rights. DRM content may be in encrypted DRM Content Format or in plaintext delivered inside a DRM message

DRM Message An OMA DRM Release 1 term defined in [DRM]

Integrity The property that data has not been altered or destroyed in an unauthorised manner. (ISO 7498-2)

Media object A digital resource e.g. a ringing tone, a screen saver, a Java game or a composite object.

Media type A MIME media type.

Permission Actual usages or activities allowed (by the Rights Issuer) over Protected Content (From [ODRL-1.1])

Play To create a transient, perceivable rendition of a resource (From [MPEG21 RDD])

Protected Content Media Objects that are consumed according to a set of Permissions in a Rights Object.

Rights Permissions and constraints defining under which circumstances access is granted to DRM content.

Rights issuer An entity who issues rights objects.

Rights Issuer An entity that issues Rights Objects to OMA DRM Conformant Devices.

Rights Object A collection of Permissions and other attributes which are linked to Protected Content.

Rights Object Acquisition Protocol (ROAP)	A protocol defined within this specification. This protocol enables devices to request and acquire Rights Objects from a Rights Issuer.
Superdistribution	A mechanism that (1) allows a User to distribute Protected Content to other Devices through potentially insecure channels and (2) enables the User of that Device to obtain a Rights Object for the superdistributed Protected Content.
User	The human user of a Device. The User does not necessarily own the Device.

3.3 Abbreviations

3GPP	3rd Generation Partnership Project
4CC	Four Character Code
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CEK	Content Encryption Key
CI	Content Issuer
CTR	Counter Mode
DCF	DRM Content Format
DRM	Digital Rights Management
HTTP	Hypertext Transfer Protocol
ISO	International Standards Organization
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messaging Service
OMA	Open Mobile Alliance
PDCF	Packetized DRM Content Format
PSS	Packet switched Streaming Service
RFC	Request For Comments
RI	Rights Issuer
RO	Rights Object
ROAP	Rights Object Acquisition Protocol
RTP	Real time Transport Protocol
RTSP	Real Time Streaming Protocol
SMS	Short Messaging Service
URI	Uniform Resource Indicator
URL	Uniform Resource Locator
URN	Uniform Resource Name

4. Introduction

OMA Digital Rights Management defines a delivery method in which the Media Object is encrypted and the Rights containing the encryption key are delivered to the Device apart from the Media Object. This specification defines the DRM Content Format for encrypted Media Objects.

The DRM Content Format is closely related to the Rights Expression Language specification [DRMREL], which defines the syntax and semantics for the Rights Objects.

4.1 Goals

~~This specification defines the DRM Content Format for encrypted Media Objects.~~ In addition to encrypting the Media Object the DRM Content Format supports metadata such as

- Original content type of the media object
- Unique identifier for this DRM protected M~~m~~edia O~~o~~bject to associate it with rights
- Information about the encryption details
- Information about the rights issuing service for this DRM protected media object
- Extensions and other media type dependent metadata

The file format is extensible, so additional features may be added later while maintaining compatibility with the older versions. Compatibility with the version 1 Content Format [DRMCF] is not maintained by this specification, thus the MIME type shall be changed as well.

There are two profiles of the Content Format. One is used for Discrete Media (such as still images) and one for Continuous Media (such as music or video).~~Furthermore, this specification includes a file profile for content of a more dynamic nature, such as music or video.~~ The profiles share data structures for the purpose of reusing components. Both profiles are based on a widely accepted and deployed standard format, the ISO Base Media File format [ISO14496-12], but the Discrete Media profile is meant to be an all-purpose format, not aiming for full compatibility with ISO media files.

The CI can decide which profile to use for their content, but in general, the profile for Continuous Media should be used for Continuous Media content, in order to create a harmonious user experience. ~~.~~The Discrete Media profile should be used for other types of content. To a User, the difference is that a DCF looks like a DRM protected file, whereas a PDCF looks and functions like a media file to the outside.

~~The DRM Content Format is closely related to the Rights Expression Language specification **Error! Reference source not found.**, which defines the syntax and semantics for the rights objects.~~

5. DRM Content Format

This section defines the DRM Content Format for Protected Content.

There are two DRM Content Format profiles:

- DCF: The first profile is used to package and protect Discrete Media. (i.e. ring tones, applications, images, etc.) The Discrete Media profile allows you to wrap any content in an envelope (DCF). That content is then encrypted as a single object agnostic of the contents internal structure and layout. This specification defines the discrete media format based on the types of the ISO base media file format [ISO-14496-12], instead of WSP types [WSP] used in Version 1 [DRMCF]. By using the ISO principles, the DCF format maintains the extensible nature of the ISO format, while keeping overhead minimal. An OMA DRM Device defined in [DRM-~~-~~v2] MUST support the DCF format as defined in this specification. In addition, version 1 DCF as defined in [DRMCF] MAY be supported.

- PDCF: The second profile is used to protect Continuous (packetized) Media (i.e. Audio and Video.) Continuous media is protected in a separate format because it is packetized. Applications that read and parse continuous media are meant to work on the file on a packet-by-packet basis. To facilitate the playback of protected continuous media, the storage format needs to be structured in such a way that the packets are individually protected. This structurally aware packetization is also required in order to stream continuous media. An OMA DRM compliant streaming server MUST be able to understand the Protected Format's structure in order to break the content into headers and packets that can be delivered to a client that understands the Protected Format.

5.1 ISO Base Media File Format

The Discrete Media profile (DCF) is ~~structurally~~ based on the ISO B~~b~~ase M~~m~~edia F~~f~~ile F~~f~~ormat data types and conventions as defined in [ISO-14496-12]. The actual data structures and conformance to the profile is defined in this specification. If a DCF includes data structures or functionalities not conforming to this specification, a compliant file parser may ignore these.

The Continuous Media profile (PDCF) is also based on the ISO base media file format, but is defined in a separate specification, in the 3GPP [TS-26.244]. By default, this specification addresses the DCF format, with an additional indication if a specified data structure is also used in the PDCF format.

~~5.1.1File Branding~~

~~The ISO base media file format allows for a file signature/brand in the file header. Files conforming to the Discrete Media profile MUST include a brand number. The file brand is 32 bits wide with the hexadecimal value `0x6F646366` (‘`odcf`’). This is preceded by a legacy version byte from DCF version 1 [DRMCF], making the file brand a total of five bytes from the beginning of the file. For files conforming to this version of the DCF specification the version value MUST be 2 (`0x02`). The ISO file type box ‘`ftyp`’ MUST NOT be used in a version 2 DCF due to its variable size.~~

~~Files conforming to the Continuous Media profile (PDCF) MUST include a file type box as specified in [TS-26.244].~~

~~5.1.2~~5.1.1 File structure

The ISO base media file format is structured around an object-oriented design of boxes. A basic box has two mandatory fields, length and type. The type identifier is used to dynamically bind a box to a statically defined type and the length is an implicit offset to the end of the box. A Box type identifier is a *Unique Identifier Number*. List of reserved numbers can be found in Appendix B. The identifier is constructed from four bytes, each representing a human-readable character, thus the name *Four Character Code* (4CC).

The ISO base format uses a language called Syntax Description Language (SDL) for defining data structures.

A basic box is defined as:

```
aligned(8) class Box (unsigned int(32) boxtype, optional unsigned int(8)[16] extended_type) {
      unsigned int(32) size;
      unsigned int(32) type = boxtype;
}
```

In files conforming to this specification, box *size* MUST be greater than 1 (e.g. *largesize* in the ISO specification not used) and the *extended*_~~-~~*type*~~*s*~~ MUST NOT be used in the mandatory boxes. Also note that in some earlier ISO specifications, the term *atom* was used to describe the file format structures, but the data structures specified in this specification SHALL be called *box*es in order to be consistent with 3GPP and current ISO specifications.

Box alignment is by default to the next byte boundary in the end of the box. Extra padding should not be needed as all datatypes in the DCF are terminated on byte boundaries.

Since one of the design goals for the DCF is extensibility, it is important to carry version information with each data type. The ISO specification has a predefined type to support this, the FullBox, which is derived from the simple Box base class.

```
aligned(8) class FullBox(unsigned int(32) type, unsigned int(8) v, bit(24) f) extends Box(type) {
        unsigned int(8) version = v;
        bit(24) flags = f;
}
```

The FullBox version is typically started from zero (0), incremented by each revision. The `flags` field MAY be used to include additional information, but SHOULD normally be set to 0, unless otherwise specified. This specification names each supported box to indicate that a box has a defined structure and a purpose in the OMA DRM ~~file~~ Content F~~f~~ormat.

There are also placeholders for extensions, with only a generic box reference. These extensions may be defined later, and thus a conforming file parser SHOULD skip any extension boxes it does not understand. In addition, all of the toplevel boxes are derived from the FullBox type, which supports version information. Later specifications MAY increment the version number if changes are made to any common data structures. Later versions of the boxes defined in this specification SHOULD remain backwards compatible with the help of this version indicator. A parser conforming to this specification MAY attempt to parse a box which has a greater version number than this specification, but the conformance is limited to the current version (0) of this specification. In any case, a conforming parser MUST support checking the version number field.

A representation of the FullBox above is:

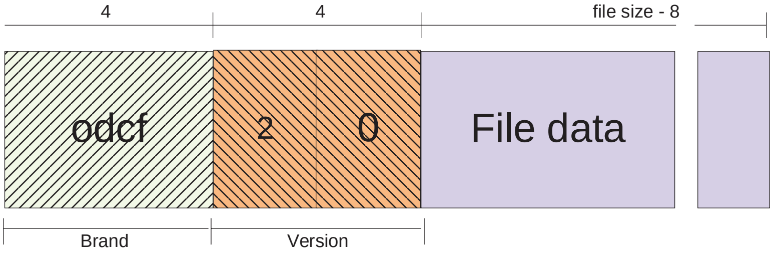
Name	Type	Value
Size	Unsigned int(32)	~~Size~~ Offset to the end of the box
Type	Unsigned int(32)	Box type ~~parameter~~4CC
Version	Unsigned int(8)	Version field
Flags	Unsigned int(24)	Additional flags

All ~~integer~~ numeric fields in the file format MUST be in network byte order.

5.1.2 File Branding

The ISO base media file format allows for a file signature/brand in the file header. Files conforming to the Discrete Media profile MUST include a brand number. The file brand is 32 bits (4 octets) wide with the hexadecimal value `0x6F646366` (‘`odcf`’). This MUST be followed by a four-octet version indicator, making the file brand a total of eight octets (64 bits) from the beginning of the file. The version field consists of a version major and minor numbers, two octets each, in network order. For files conforming to this version of the DCF specification the version value MUST be 2.0 (`0x00020000`). A conforming file parser MUST support the version number. The Figure 1 shows the relationship of the file brand, version and rest of the file content.

Figure 1~~1~~: DCF file header and body



The ISO file type box '`ftyp`' MUST NOT be used in a version 2.0 DCF due to its variable size. Files conforming to the Continuous Media profile (PDCF) MUST include a file type box as specified in [TS26.244].

5.2 Common Boxes

5.2.1 The Common Headers Box

```
aligned(8) class OMADRMCommonHeaders extends FullBox('odhe', version, 0) {
      unsigned int(16)      EncryptionMethod;      // Encryption method
      unsigned int(16)      EncryptionPadding;     // Padding type
      unsigned int(32)      PlaintextLength;       // Plaintext content length in bytes
      unsigned int(16)      ContentIDLength;       // Length of ContentID field in bytes
      unsigned int(16)      RightsIssuerURLLength; // Rights Issuer URL field length in bytes
      unsigned int(16)      Textual~~Extended~~HeadersLength; // Length of the ~~ExtendedHeaders~~
TextualHeaders array in bytes
      char                  ContentID[];           // Content ID string
      char                  RightsIssuerURL[];     // Rights Issuer URL string
      string                ~~ExtendedHeaders~~TextualHeaders[];    // Additional headers as Name:Value
pairs
      Box                   ExtendedHeaders[];     // Extensible headers, to the end of the box
}
```

The Common Headers box defines a structure for the required headers. This box MUST appear in both DCF and PDCF. This box includes the mandatory headers as fixed fields and provides a mechanism to insert additional headers as arbitrary name value pairs. For application in DCF and PDCF, see sections 5.3.3.2 and 5.4.2.2.5~~5.4.2.2.5~~ for details.

A D~~d~~evice SHOULD NOT edit any of the fields in the Common Headers box.

5.2.1.1 Common Headers Version

The *version* field of the FullBox defines which version of DRM Content Format specification was used by the author of the content object. The value for *version* MUST be 0 for objects conforming to this specification.

5.2.1.2 EncryptionMethod Field

The *EncryptionMethod* field defines how the encrypted content can be decrypted. Values for the field are defined in the table below.

Table 1. Algorithm-id values

Algorithm-id	Value	Semantics
NULL	0x0000	No encryption for this object

AES_128_CBC	0x0001	AES symmetric encryption as defined by NIST. 128 bit keys. Cipher block chaining mode (CBC). 128 bit initialization vector prefixing the ciphertext. Padding according to RFC 2630, unless overridden by the *PaddingScheme* field.
AES_128_CTR	0x0002	AES symmetric encryption as defined by NIST. 128 bit keys. Counter mode (CTR). 128 bit IV is constructed using a unique counter that prefixes the ciphertext.

Rights Issuers MUST take care in using NULL EncryptionMethod because, given a null-encrypted element within a DCF, the following statements hold true:

- Null-encrypted elements do not have any Confidentiality protection.
- Null-encrypted elements can be used without an associated Rights Object.
- Null-encrypted elements may not have any integrity protection, because the hash for integrity check is included in the associated Rights Object.

5.2.1.2.1 PaddingScheme Field

The *PaddingScheme* parameter defines how the last block of ciphertext is padded.

Values of the PaddingScheme field are defined in the table below:

Table 2. PaddingScheme values

Padding-Scheme	Value	Semantics
NULL	0x0000	No padding. This padding-scheme MUST only be used if the *PlaintextLength* parameter is greater than zero.
RFC_2630	0x0001	Padding according to RFC 2630. If this padding scheme is used, PlaintextLength MUST be zero.

5.2.1.3 PlaintextLength Field

The *PlaintextLength* field defines the length of the original plaintext. Some simple padding schemes may require that the plaintext length is explicitly defined. If the field is not used, it MUST be set to zero.

5.2.1.4 ContentIDLength Field

The *ContentIDLength* field defines the number of bytes occupied by the *ContentID* field.

5.2.1.5 RightsIssuerURLLength Field

The *RightsIssuerURLLength* field indicates the number of bytes occupied by the *RightsIssuerURL* field.

5.2.1.6 ~~ExtendedHeaders~~TextualHeadersLength Field

The ~~ExtendedHeaders~~*TextualHeadersLength* field indicates the number of bytes occupied by the ~~ExtendedHeaders~~*TextualHeaders* field. Although it is possible with this version of the parent box ~~allows~~ to implicitly determine~~ning~~ the ~~ExtendedHeaders~~TextualHeaders field length from the box length, this might not be the case in future versions. Thus, conforming tools MUST use the ~~ExtendedHeaders~~TextualHeadersLength field.

5.2.1.7 ContentID Field

The *ContentID* field MUST contain a unique identifier for this DRM protected content object. The value MUST be associated with a CEK. The value MUST be encoded using US-ASCII encoding.

The value MUST be a URI according to [RFC2396]. It is the responsibility of the content author to guarantee the uniqueness of the ContentID. URI schemes like “cid:local-part@domain” as defined in [RFC2392] MAY be used.

If the content object is referenced from a DRM rights object, the value of the ContentID field MUST match the value of the referencing element of the rights object as defined in [DRMREL].

5.2.1.8 RightsIssuerURL Field

The *RightsIssuerURL* field defines the Rights Issuer URL. The Rights Issuer URLs MAY be used by the consuming device to obtain rights for this DRM protected content object. The mechanism is defined in OMA DRM specification [DRM-v2]. The value of the RightsIssuerURL field MUST be encoded using US-ASCII encoding. The length of this field is indicated by the RightsIssuerURLLength field.

The value of the RightsIssuerURL MUST be a URL according to [RFC2396].

5.2.2 Extended Headers

There are two mechanisms to extend the mandatory header information. The ~~*ExtendedHeaders*~~*TextualHeaders* and *ExtendedHeaders* fields MAY contain ~~headers defining~~ additional information about the content.

~~The~~Textual headers are represented by name value pairs, where name and value are separated with a colon ‘:’ and the pair is terminated with a NULL character. A header (name value pair) MUST NOT include leading or trailing whitespace (such as \r\n). Further, a header name MUST NOT include a colon (‘:’) character, as the first instance or the character will stop scanning for the header name. Header value MAY include colon characters as the value is always assumed to continue after the first colon until a NULL character is reached.

The next header name MUST begin immediately after the terminating NULL character of the previous header, if ~~ExtendedHeaders~~TextualHeadersLength is greater than the current scanning position. All headers MUST have a value, i.e. an empty value is not permitted.

The extended headers field continues until the TextualHeadersLength offset or the end of the box is reached. ~~NOTE: Any future versions of the common headers box are not restricted by this requirement, and thus t~~The ~~ExtendedHeaders~~TextualHeadersLength field MUST be used to determine the ~~ExtendedHeaders~~TextualHeaders field length.

An example ~~textual~~ representation of the extended textual headers:

```
Content-Vendor:GreatCompany\0Icon-URI:http://www.greatcompany.com:8080/contenticon.png\0
```

Each supported header is defined using augmented Backus-Naur Form (BNF) [RFC2234]. The extended headers are encoded using UTF-8 encoding.

The *ExtendedHeaders* array is used for future additions, and it can nest e.g. binary data. The array MAY have zero or more sub-boxes, and it spans until the end of the `OMADRMCommonHeaders` box is reached. The TextualHeaders field SHOULD NOT be used for large strings, such as encoded binary data, the ExtendedHeaders field SHOULD be used instead. A Device MAY ignore the extended headers it does not support.

5.2.2.1 Silent header

The *Silent* header is an indication to the client that the Rights Object for this DCF can be obtained silently from the Rights Issuer, without user interaction for payments, etc. The device MAY use this header to determine how to acquire the Rights Object.

```
Silent := “Silent” “:” silent-method “;” parameter
silent-method := token
parameter:= silent-rights-url
```

silent-method	Semantics
“on-demand”	Rights should be acquired silently, on demand when the user chooses to play the content.
“in-advance”	Rights should be acquired in advance, opportunistically.

The parameter `silent-rights-url` MUST be a URL according to [RFC2396].

The parameter `silent-rights-url` MUST be specified on the Silent header. The device MUST use this `silent-rights-url` to obtain rights silently and automatically.

If this request cannot be reconciled to a prior purchase transaction, the RI server MUST return an error. The client can take further action based on this error indication. It is recommended that the client start a browsing session with the RI URL if the context is a user-initiated session. If the context is a DRM-agent initiated session to acquire rights silently and automatically, then it is better for the client to abandon the rights acquisition effort.

5.2.2.2 Preview header

The *Preview* header contains an indication to the client that it is possible to provide a preview for this DCF.

If the `preview-method` is “`instant`”, then the specific media element to be used for preview MUST be indicated using the `preview-element-uri` parameter. In addition, this media element MUST be NULL-encrypted, and as such, MUST have an `EncryptionMethod` header with the `algorithm-id` parameter set to `NULL`.

```
Preview := “Preview” “:” preview-method *(“;” parameter )
preview-method := token
parameter :=  preview-element-uri [“;” preview-rights-url]
```

Preview-method	Semantics
“instant”	This indicates that one of the elements within this composite object can be used for preview. If `instant` method is specified, then `preview-element-uri` MUST be specified.
“preview-rights”	This indicates that a preview Rights Object can be obtained by requesting it silently from the Rights Issuer, without user interaction If `preview-rights` method is specified, then `preview-rights-url` MUST be specified.

The parameter `preview-element-uri` MUST be a unique identifier and a URI according to RFC2396. And, it MUST resolve to an element present within the DCF.

The parameter `preview-rights-url` MUST be a URL according to RFC2396.

If the preview-method is indicated as “instant”, the preview element can be used freely with unlimited use, without acquiring any Rights Objects.

If the preview-method is “`preview-rights`”, then the `preview-rights-url` MUST be indicated as a parameter. When the client connects to the Rights Issuer with this URL, the result is either a Rights Object or an error. This MUST NOT result in any re-direction.

5.2.2.3 ContentName header

The *ContentName* header contains a descriptive name for this DRM protected content object. The name is only informative and the device MAY use it e.g. to derive a filename when the DRM protected object is received and stored into a local repository. Other names may be transmitted outside this object (e.g. Content-Disposition header in HTTP) and they may override the name specified in this element.

```
ContentName := “Content-Name” “:” token
```

5.2.2.4 ContentDescription header

The *ContentDescription* header contains a description of the DRM protected content object. This text is informative and the device MAY display it to the user prior to using the RightsIssuer field.

```
ContentDescription := “Content-Description” “:” token
```

5.2.2.5 ContentVendor header

The *ContentVendor* header contains a textual string representing the name of the organisation that provided the media object. This text is informative and the device MAY display it to the user prior to using the Rights Issuer URL field.

```
ContentVendor := “Content-Vendor” “:” token
```

5.2.2.6 IconURI header

The *IconURI* header contains a URI where an appropriate icon for this content may be retrievable from. The device MAY use this header to request the object at this URI, and if an appropriate content is returned, use this as an icon associated with the content to the user.

The value of the IconURI MUST be a URI according to RFC2396~~**Error! Reference source not found.**~~.

```
IconURI := “Icon-URI” “:” token
```

5.2.2.7 Unsupported headers

Content author MAY insert additional headers to the ~~*ExtendedHeaders*~~*TextualHeaders* field. Additional headers MUST follow the generic syntax defined below, encoded using UTF-8 encoding.

```
OtherHeader := Header-name “:” Header-value
Header-name := token
Header-value := token
```

Consuming Devices MUST ignore the headers that they do not recognize.

5.3 Discrete Media Format

5.3.1 DCF MIME Type

The MIME type for objects conforming to the format defined in this section MUST be

```
application/vnd.oma.drm.content.v2?
```

5.3.2 DCF File Format

The structure of the Discrete Media profile of DRM protected content (DCF) MUST be according to the structure definitions below. The file brand of eight~~five~~ octets MUST precede the first box.

A DCF file MUST include at least one `OMADRMContainer` box. The `OMADRMContainer` box is a container for a single content object and its associated headers. It MUST appear on the top level, i.e. to conform to this specification, it MUST NOT be nested inside another data type. There MAY exist multiple `OMADRMContainer` boxes in a file, but one MUST immediately follow the file brand, and they MUST all be on the top level in the nesting structure.

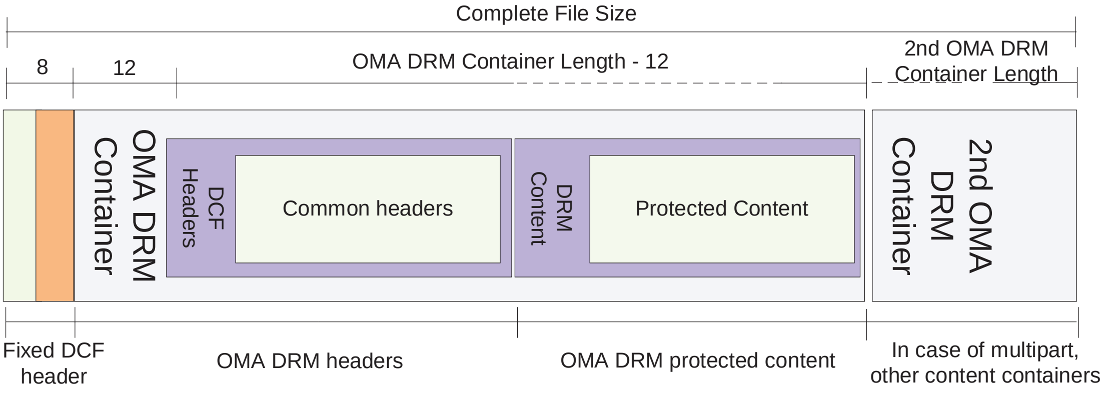
The *version* indicator field in each box MUST be 0 for files conforming to this specification.

5.3.3 Overall structure

The high-level overview of the DCF format is depicted in the Figure 2. The mandatory parts of the format include the file header with Brand number and Version fields, immediately followed by an OMA DRM Container box. The OMA DRM Container box MUST include a DCF headers box and a Proctected Content box.

The design principles for the format include that the DCF headers box is located at a fixed offset from the beginning of the file, and thus, the OMA DRM Container box MUST be the first box after the file header of eight octets and the DCF headers box MUST be the first box in the OMA DRM Container.

Figure 2~~2~~: DCF structure



The table below outlines the mandatory boxes and their order. Additional boxes MAY be added after the mandatory boxes have first appeared. Table 3 shows the nesting order of the mandatory boxes, on the left is the parent and on the right, the child.

Table 3: Logical DCF box structure diagram

Data type/value			Nesting level	Offset from beginning of file	Field purpose
‘odcf’			0	0	File header magic (4 bytes)

0x00020000			0	4	File version (2 bytes major, 2 bytes minor)
Box(‘odrm’)			0	8	OMA DRM container box
	Box(‘odhe’)		1	20	DCF headers box
		Box(‘ohdr’)	2	36 + ContentTypeLength	OMA DRM common headers box
	Box(‘odda’)		1	20 + Box(‘odhe’)	OMA DRM content data box
Box(‘odrm’)			0	8 + Box(‘odrm’)	If multipart DCF, additional OMA DRM container box

~~5.3.2.1~~5.3.3.1 OMA DRM Container Box

```
aligned(8) class OMADRMContainer extends FullBox('odrm', version, 0) {
      OMADRMDiscreteHeaders  ContentHeaders;        // Headers for discrete DCF
      OMADRMContentObject    DRMContent;            // Actual encrypted content
      Box                    Extensions[];          // Extensions, to the end of the box
}
```

The OMADRMContainer box MUST include a single OMADRMHeaders box and a single OMADRMContent box, followed by optional extensions. The Extensions inside the OMADRMContainer box are defined by OMA.

~~5.3.2.2~~5.3.3.2 Discrete Media Headers Box

```
aligned(8) class OMADRMDiscreteHeaders extends FullBox('odhe', version, 0) {
      unsigned int(8)        ContentTypeLength;     // Content Type Length
      char                   ContentType[];         // Content Type String
      OMADRMCommonHeaders    CommonHeaders;         // Common headers (same as with PDCF)
}
```

The Discrete Media profile headers box includes fields specific to the DCF format and the common headers box. There MUST be exactly one OMADRMDiscreteHeaders box in a single OMA DRM Container box, as the first box in the container.

The *ContentType* field indicates the actual media type contained in the OMA DRM container. In addition, the discrete headers box includes the common headers box. There MUST be exactly one OMADRMCommonHeaders (see section 5.2.1 for details) box per a single OMADRMDiscreteHeaders box.

Table 4. OMA DRM discrete media header fields

Field name	Type	Purpose
ContentTypeLength	Unsigned int(8)	Length of the ContentType field
ContentType	ContentTypeLength octets	The MIME media type of the plaintext data encoded as US-ASCII

~~5.3.2.2.1~~5.3.3.2.1 ContentType

The *ContentType* field MUST indicate the original MIME media type of the DRM protected content i.e. what content type the result of a successful decryption of the `OMADRMContent` box represents. The *ContentType* field is encoded using US-ASCII encoding and MUST NOT include a NULL character.

~~5.3.2.2.2~~5.3.3.2.2 CommonHeaders

The *CommonHeaders* field MUST be the same box as defined in 5.2.1.

~~5.3.2.3~~5.3.3.3 Content Object Box

```
aligned(8) class OMADRMContentObject extends FullBox('odda', version, 0) {
      unsigned int(32) OMADRMDataLength;    // Length of the encrypted content
      bit(8)  OMADRMData[];                 // Encrypted content
}
```

The Content Object box MUST include only the data length field and data bytes for a single Protected Content Object. Later revisions of this box may include additional fields, so conforming implementations MUST use the OMADRMDataLength field to indicate/determine the amount of actual data bytes.

Table 5: Content Object box

Field name	Type	Purpose
OMADRMDataLength	Unsigned int(32)	Length of the OMADRMData field, in octets
OMADRMData	bit(8) []	Protected Content bytes, as specified by the OMADRMDiscreteHeaders box

~~5.3.2.4~~5.3.3.4 Extended Boxes

Any additional boxes contained in a single OMA DRM container box have not been defined yet.

~~5.3.3~~5.3.4 Multiple OMA DRM Containers

A DCF MAY include more than one OMA DRM Container. Each of these containers MUST conform to the definition of the OMA DRM Container, and ~~must~~ MUST be placed sequentially on the top level (i.e. nesting them is not allowed).

Each OMA DRM Container MUST have a unique ContentID in its headers. This kind of a DCF with multiple Protected Content containers SHALL be called a Multipart DCF.

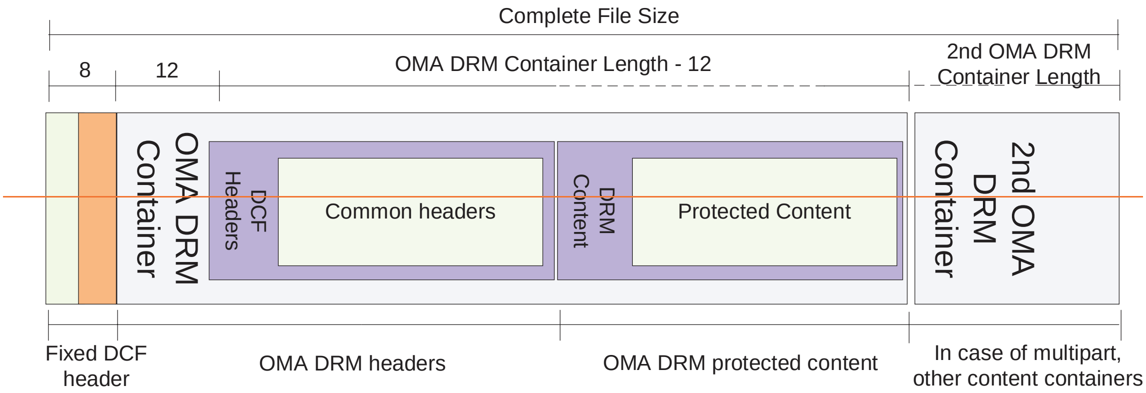
Note that a Multipart DCF is different from a DCF including a Composite Object. For Composite Objects (such as MIME multipart, ZIP and so on), there exists only one set of OMA DRM headers, and only one Rights Object, whereas for Multipart DCFs, there are separate headers for each object in the Multipart DCF, and support for having different Rights for Content Objects.

~~5.3.4~~5.3.5 Metadata Support

Additional proprietary extension boxes MAY be added after the first OMA DRM Container. A conforming file parser, which does not recognize the additional boxes, MUST ignore them. However, any extensions MUST be designed in a way that the mandatory parts of this specification are always included and the file remains interoperable with conforming implementations.

An example of metadata could be adding a box for ID3 tag or an RDF document to describe the content. The identifiers or exact content are not specified in this specification~~ (TODO)~~.

TODO: should we define a common format for media-specific metadata box? Like Box('meta')



5.4 Packetized Media Format (PDCF)

The Continuous (Packetized) Media profile is targeted for media content like audio and video. Audio and video files MAY be included in a DCF format, but for creating a consistent user experience for OMA DRM, the PDCF format SHOULD~~should~~ be used for continuous media types like audio and video.

5.4.1 PDCF MIME Type

The MIME type for objects conforming to the format defined in this section MUST be

video/3gpp or audio/3gpp

The internal file branding and structure must conform to the specification [TS~~-~~26.244]. The PDCF format MAY be used for downloaded content or for hosting streamable content. The format is limited to the media types listed in the 3GPP specification.

An audio/3gpp file is an instance of a video/3gpp file containing only audio tracks. This specification will support both, but for clarity, only use video/3gpp to refer to the 3GPP media file format [TS26.244].

5.4.2 PDCF File format

The PDCF file format is a sub profile of the video/3gpp format, which is used for (downloaded) protected media content. The structure and conformance to the profile are not defined in this specification, but instead, this specification defines the OMA DRM key management part of the format. The video/3gpp~~3GP~~ file format allocates space for a "black box" describing the key management governing access to the media content. In a PDCF file, this box MUST be the `OMADRMKMSBox`.

The protected video/3gpp file format data structures are defined by the 3GPP, but this specification gives an overwiev of the data structures and presents their OMA DRM aspects. Other DRM mechanisms MAY be used in video/3gpp files supporting DRM, but not in PDCF files, as explained in this specification.

5.4.2.1 Original Sample Entries

Each packetized track in a PDCF file has a corresponding *Sample Entry*. The Sample Entry includes information about the bitstream, such as what resolution, codec etc. were used to make the encoding transform to the track, and how to make a reverse transform to reconstruct e.g. video to the display.

The original sample entries describing the track are replaced with a derived type, which hides the codec used to encode the corresponding track. This is done to make the sample format incompatible with non DRM aware Devices, and thus avoid any bad consequences, such as crashing the media player, while opening the media. A `ProtectionIfoBox` is appended to the derived sample entry and the extended sample entry box only indicates that an encrypting "codec" was used to encode the content. The actual codec, along with DRM specific parameters, is indicated in the `ProtectionInfoBox` and its child boxes.

For example, a video track sample entry is derived from the `VisualSampleEntry`, and the codec type is replaced with an encryption indicator '`encv`'. The `ProtectionInfoBox` containing the original codec identifier is appended to the box.

```
class EncVisualSampleEntry(codingname) extends VisualSampleEntry ('encv'){
      ProtectionInfoBox(codingname) info;
}
```

5.4.2.2 Protection Information

The `ProtectionInfoBox` is used in video/3gpp files to indicate that a track is DRM protected, and to carry information about the protection scheme. The `ProtectionInfoBox` is a container box.

The `OriginalFormatBox` is used to indicate the actual codec used, `SchemeTypeBox` to indicate the DRM key management system and `SchemeInformationBox` for passing key management system specific information. All of these boxes MUST appear in a `ProtectionInfoBox` as below, unless otherwise specified in [TS26.244].

```
aligned(8) class ProtectionInfoBox(fmt) extends FullBox('sinf', 0, 0) {
      OriginalFormatBox(fmt)          original-format;
      SchemeTypeBox                   scheme-type;
      SchemeInformationBox            info;
}
```

5.4.2.2.1 DRM Scheme Type

The `ShemeTypeBox` includes information on which DRM system is being used to manage keys and decryption of the content. A video/3gpp file MAY support also other key management systems than OMA DRM, the key management system in use is indicated by a 4CC in the *scheme_type* field.

For PDCF files conforming to this specification, the *scheme_type* MUST be the 4CC 'odkm', and *scheme_version* MUST be 0x0200 (version 2.0). If OMA DRM key management scheme 'odkm' is indicated, then the video/3gpp file is a PDCF and MUST contain at least one `OMADRMKMSBox`. A PDCF SHALL support only OMA DRM for the key management system.

```
aligned(8) class SchemeTypeBox extends FullBox('schm', 0, flags) {
      unsigned int(32)                scheme_type;          // 4CC identifying the scheme
      unsigned int(16)                scheme_version;       // scheme version
      if (flags & 0x000001) {
             unsigned int(8)          scheme_uri[];         // browser uri
      }
}
```

5.4.2.2.2 Scheme Information

The `SchemeInformationBox` is used to carry DRM key management system specific information, thus it is only a container box. For OMA DRM, this box MUST include exactly one `OMADRMKMSBox,` as the first box in the array.

```
aligned(8) class SchemeInformationBox extends FullBox('schi', 0, 0) {
      Box    scheme-specific-data[];
}
```

5.4.2.2.3 OMA DRM Key Management System

There MAY be several instances of the `OMADRMKMSBox` in a PDCF file, and one can appear either at the toplevel *movie box* or exactly one per each protected track. There MUST NOT be key management boxes in both movie level and track level. The exact locations for these boxes are defined in the 3GPP file format specification [TS26.244].

```
aligned(8) class OMADRMKMSBox extends FullBox('odkm', version, 0) {
      OMADRMSampleFormatBox           sample_format;
```

```
       OMADRMCommonHeaders           headers;
}
```

Contained in the OMADRMKMSB~~b~~ox there MUST be a single OMADRMS~~s~~ample~~ ~~F~~f~~ormatB~~ b~~ox and a single OMA DRM Common~~ ~~Headers box. The sample format box is used to indicate the format of the payload headers placed on media access units. The internal structure of the access units might not make any sense to a Device that is not OMA DRM aware, but an OMA DRM aware Device will look at the sample format box to know how to extract the protected content.

~~5.4.2.1~~5.4.2.2.4 Sample Format

The *Sample Format* specifies the format for each ~~network~~ access unit, more specifically the payload header type for OMA DRM protected content when used with a streaming protocol such as the 3GPP PSS [TS~~-~~26.234].

```
aligned(8) class OMADRMSampleFormatBox extends FullBox('osfm', 0, 0) {
       bit(1)                selective-encryption;
       bit(7)                reserved;
       unsigned int(8)       key-indicator-length;
       unsigned int(8)       IV-length;
}
```

For more information on the access unit format, see section 5.4.3.1.

~~5.4.2.2~~5.4.2.2.5 Common Headers

The Common headers box is exactly the same as defined in section 5.2.1.

5.4.3 PDCF Streaming format

Streaming PDCF content is leveraging the protected video/3gpp file format, and widely deployed standard streaming protocols. This specification uses the 3GPP PSS service protocols [TS26.234] as a reference, but the encrypted payload wrapper format MAY be used in any other streaming service using RTSP streaming, SDP signaling and RTP transport.

Supporting the PDCF streaming is OPTIONAL, even if PDCF format is supported. A multimedia streaming session MAY consist of PDCF streamed tracks and unprotected tracks.

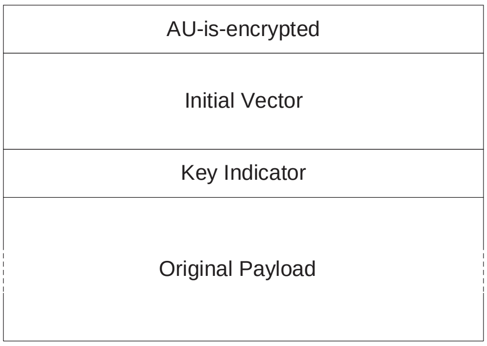
Streaming protected tracks is signaled through SDP parameters, using information contained in the sample format entries of the protected video/3gpp file. A streaming server derives network packets from a *hint track* in the media file.

Although the streaming format and payload are defined in [TS~~-~~26.244], they are explained here from the OMA DRM perspective.

5.4.3.1 RTP Payload

The RTP payload format consists of two parts: the encrypted payload wrapper and the actual media payload. The media payload (e.g. H.263 video) is packetized according to the appropriate standard. The encrypted payload wrapper includes a header with additional signaling information, such as selective encryption indicator and initial vector for the packet. With this mechanism, one encrypted payload specification is used to protect any standard RTP payload. Also a benefit of the wrapper format is that the DRM system is fully functional in networks supporting basic RTP profiles, and thus not placing requirements on existing network configurations.

Figure 3: Encrypted wrapper payload format



The DRM sample format in section 5.4.2.2.4 is used to signal the format for the encrypted payload wrapper. The fields in the wrapper format are prefixing the actual protected payload and to the RTP protocol layer, it only looks like it is transporting the wrapper payload media type.

Table 6: Encrypted Payload Wrapper fields

Parameter name	**Purpose**
AU-is-encrypted	Boolean encryption indicator (0=false, 1=true).
Initial Vector	IV for the access unit
Key Indicator	Key indicator for CTR mode

~~Add more details as 3GPP finalizes the payload format~~

Note: 3GPP has not yet specified mechanisms for real-time transport of encrypted PSS media, e.g. an encrypted wrapper payload format as described in this section. This specification will be updated according to the 3GPP specifications when the relevant parts are available.

5.4.3.2 Hint Tracks

The video/3gpp format supports special hint tracks for streaming servers. Hint tracks include pointers to network packets within the packetized file. Using these pointers, the streaming server is able to stream the file even without knowledge of what the packets actually include. Downloaded PDCFs MAY include hint tracks, but in normal playback of downloaded content, they are ignored.

5.4.3.3 Session signaling

For PDCF streaming, the session descriptors (SDP files) MUST include information about the wrapper payload. The format parameters for the wrapper format are used to signal e.g. DRM key management parameters.

The generic parameters are defined in [TS26.234]. In the *Encryption Parameters*, PDCF streaming MUST support the AES 128 cipher in counter mode. If the *Selective Encryption* feature is disabled for a track, the Device MUST discard all packets belonging to this track where the encryption indicator is zero (unencrypted).

The *Key Management Specific* parameters MUST include the mandatory OMA DRM headers, as name value pairs. These parameters MUST be derived from the key management box in PDCF.

Table 7: Required OMA DRM specific parameters

Parameter name	**Purpose**
ContentID	ContentID for the protected track
RightsIssuerURL	The RightsIssuerURL for fetching rights

Other headers MAY be added to the key management specific parameters, and a consuming Device MUST pass them to the DRM agent. The DRM agent will then act accordingly and acquire rights for the stream as appropriate. The semantics of the headers are the same as the common headers defined in section 5.2.

Appendix A. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [CREQ].

Item	Function	Reference	Status	Requirement
DRM-DCF-GEN-1	AES128CBC encryption algorithm	5.2.1.2	M	
DRM-DCF-GEN-2	RFC 2630 padding scheme	5.2.1.2.1	M	
DRM-DCF-GEN-3	NULL padding scheme	5.2.1.2.1	M	
DRM-DCF-GEN-4	PlaintextLength field	5.2.1.2.1	M	
DRM-DCF-GEN-5	RightsIssuer field	5.2.1.7	M	
DRM-DCF-GEN-6	ContentName header	5.2.2.3	O	
DRM-DCF-GEN-7	ContentDescription header	5.2.2.4	O	
DRM-DCF-GEN-8	ContentVendor header	5.2.2.5	O	
DRM-DCF-GEN-9	IconURI header	5.2.2.6	O	
DRM-DCF-GEN-10	Ignore unsupported headers	5.2.2.7	M	
DRM-DCF-GEN-11	AES128CTR mode encryption algorithm	5.2.1.2	O	Mandatory if PDCF is supported? Or is this always used for PDCFs?
DRM-DCF-GEN-12	DCF support	5.3	M	
DRM-DCF-GEN-13	PDCF support	5.4	O	
DRM-DCF-GEN-14	Ignore unsupported boxes in DCF		M	
DRM-DCF-GEN-15	Check DCF brand	5.1.2	M	
DRM-DCF-GEN-16	DCF version		M	
DRM-DCF-GEN-17	FullBox version	5.1.1	M	
DRM-DCF-GEN-18	64 bit box length	5.1.1	?	
DRM-DCF-GEN-19	PDCF streaming	5.4.3	O	

Appendix B. Reserved Numbers (Informative)

Table 8: Reserved identifier constants in the DCF format

UUID	Reference	Purpose
‘odcf’	1.1.1	File brand
‘odrm’	5.3.3.1	OMA DRM Container box
‘ohdr’	5.2.1	Common headers box
‘odhe’	5.3.3.2	Headers box for the Discrete Media profile box
‘odda’	5.3.3.3	Protected Content box

Table 9: Reserved OMA DRM specific identifier constants in the PDCF format

UUID	Reference	Purpose
‘odkm’	5.4.2.2.2, 5.4.2.2.3~~5.3.2.1~~	OMA DRM ~~Container~~ scheme type, OMA DRM scheme information box identifier~~box~~
‘ohdr’	0~~5.2.1~~	Common headers box

Appendix C. Change History (Informative)

A.1. Approved Version History

Reference	Date	Description
n/a	n/a	No prior version –or- No previous version within OMA

A.2. Draft/Candidate Version <current version> History

Document Identifier	Date	Sections	Description
Draft Versions OMA-DRM-DCF-V2_0~~1_2~~	01 Oct 2003	Initial draft	Moved to new format and incorportated input from Sami
	03 Nov 2003		Removed DCF legacy version, added support for binary headers, major update of the PDCF sections
Candidate Version OMA-DRM-DCF-V1_2		n/a	Status changed to Candidate by TP TP ref # OMA-TP-2003-0abc-CandidateRequest_xxyyz_V1_2

3GPP TSG SA WG3 Security — S3#31 **S3-030752**

18 - 21 November 2003

Munich, Germany

Title:	**DRM usage for MBMS security**
Source:	**Nokia**
Document for:	**Discussion and decision**
Agenda Item:	**6.20**
Work Item:	**MBMS**

1 Introduction

SA3 has received liaisons from SA4 on work division between 3GPP and OMA on DRM protected content [S3-030313] and suitable cipher selection [S3-030314]. The response liaison from SA3#30 Porto meeting to SA4, OMA-SEC and OMA-DRM+DL [S3-030650] states*"SA3 is considering solutions for the encryption and integrity protection of MBMS streaming media and it would be advantageous to consider alignment of these solutions (and the associated requirements) with the encryption and integrity protection mechanisms for DRM."*

This paper studies how OMA DRMv2 could be used in the MBMS context. Ericsson has also discussed multicasting DRM content via the MBMS architecture in SA3#28 contribution [S3-030248].

2 Discussion

2.1 OMA DRMv2

Let us first recall that the main idea of MBMS security is to keep unauthorized service listeners out. The basic target of OMA DRM is very similar.

OMA DRM model is not about absolute security but agreeing suitable ciphering method for the content to be pre-protected. The majority of the content is assumed to be off-line content. OMA DRM has support for both standard 3GPP streaming and download. A protected 3GPP SA4 3GP file format is defined for PSS and that is applied for downloadable and streamed content.

The below Figure 1 presents the parts of a Media DRM system according to the OMA DRMv2 view. This figure has been presented in SA4 contribution [S4-030367].

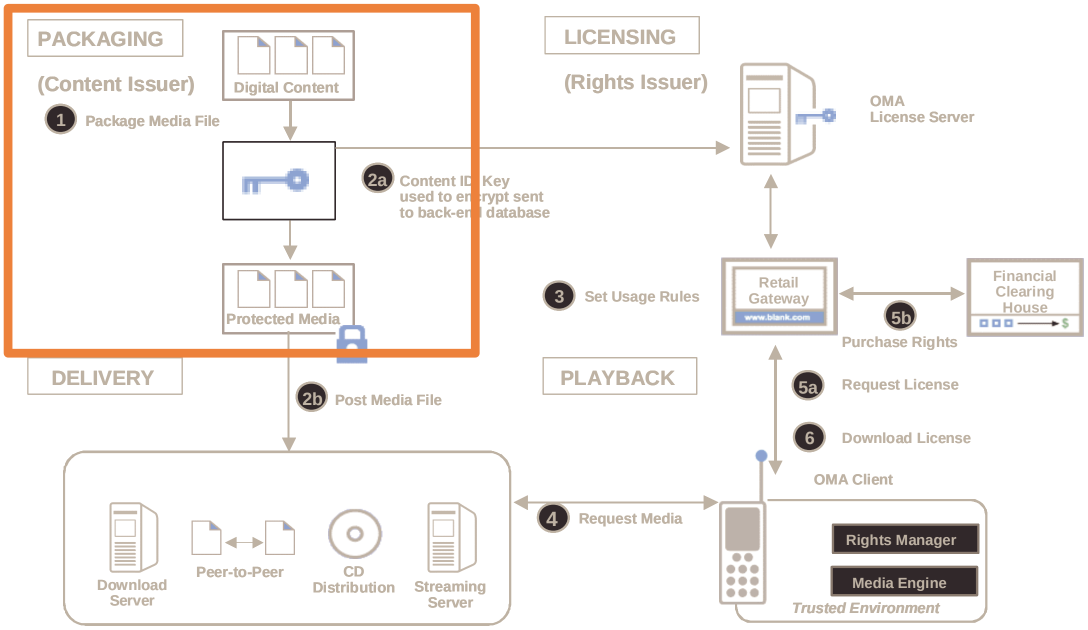


Figure 1 – OMA DRMv2 view

2.2 OMA DRMv2 usage in MBMS

Taking into account the background given in earlier chapter, OMA DRM could be used to support the MBMS service. The OMA DRM model assumes that all rights have to be managed for contents (but it may well be that the same rules apply for every piece of content, and different rules are not needed for every content). One difficulty in aligning DRM model with MBMS is that MBMS does not have a guaranteed delivery for the content, as there is no way of knowing which UEs have received the data. Hence, it may happen that UE has purchased rights for certain content but has not received the content itself.

One potential way of applying DRM model better in MBMS context is described in the following. BM-SC could act in the role of a content owner in cases where the content provider does not support DRM or it wants to delegate the role of the content owner to the BM-SC for some other reason. Now DRM could be used for every content, but we would not have a complete solution for the reason of explained above about the nonguaranteed content delivery. On the other hand, this is an open issue to be solved also for non-DRM protected content. For instance, SRTP as such doesn't have support for the reliable delivery. Anyway, MBMS BM-SC could have certain functionalities of OMA DRMv2 entities, like delivery and packaging, and the 3GPP MBMS needs to define the missing parts. With this approach, i.e.if BM-SC would be DRM compliant entity (i.e. rights issuer combined with packaging and delivery), then we would have the benefit that DRM mechanisms could be fully used for the protection of the content and need for MBMS-specific solutions would be minimized.

The division of functions between the MBMS-specific part and generic OMA DRM-based part would be the following. For MBMS service you have to have a subscription. Subscription management would be MBMS specific and the associated security mechanims, e.g. authentication of the subscriber at the highest level (for joining the service) have to be defined in 3GPP. This would be done by using the UICC. As a byproduct, we obtain shared secrets between BM-SC and UE. However, it is FFS how the DRM mechanisms could then utilize these secrets.

3 Conclusions

Principles were presented by which the SA3 work on MBMS security can be aligned with the ongoing co-operation between OMA DRM group and SA4 group for PSS.

4 References

[S3-030248] Authentication in MBMS, Discussion paper in SA3#28, May 2003, Ericsson

[S3-030313] LS (from SA WG4) on DRM Content Format, SA3#29 San Francisco

[S3-030314] Reply LS (from SA WG4) to “Reply to Liaison Statement on MBMS Codec Requirements”, SA3#29 San Francisco

[S3-030650] Reply LS on cipher suite for DRM-protected streamed media for PSS, SA3#30 Porto

[S4-030367] DRM Content Format, SA4#26, May 2003