3GPP TSG SA WG3 Security — S3#33 **S3-040231**
10 - 14 May 2004
Beijing, China

Source: **Ericsson**

Title: **Discussion paper on MBMS data protection for download**

Document for: **Discussion/Decision**

Agenda Item: **MBMS**

1 Introduction

Very little work has been done in SA3 on the download case for MBMS. Hence this document tries to progress the work. The aim of this document is to study some aspects of, and possible solutions for the protection of download in MBMS. As the conclusions of the document shows, there are few directions that will lead to a security solution that will be ready in a R6 time frame. Since it is the wish of operators to have download in R6, it is important to agree on a solution that will be ready in time.

Through out the document it is assumed that the data is transmitted over FLUTE/UDP/IP [2].

2 Channel Protection vs. Content Protection

As was argued in S3-020533 [4], IPsec has some drawbacks (e.g., the need for PKI infrastructure and obstruction of header compression) that make it unsuitable for protection of download in MBMS. Therefore IPsec will not be considered in this document. There are basically two ways in which the MBMS data can be protected on the application layer. The first being protection of the actual object, and the second being protection of the channel over which the object is transmitted.

The MBMS service is about delivering content to the ME from the BM-SC, i.e., MBMS offers a broadcast/multicast channel through which objects can be delivered. Hence there is no need to define a protection mechanism that covers more than the actual channel. In other words, once the object is delivered to the ME, it is no longer the responsibility of the MBMS service to assert the security of the object. There are other solutions for that.

The MBMS service should not be concerned with the structure or semantics of the objects delivered over the channel; they should be viewed as opaque. There is however one exception to the ignorance of the semantics of the object, as will be discussed further in Section 3. The exception is that it will be beneficial for the service to be able to avoid the overhead of protection when the object to be delivered is already protected, e.g., during delivery of a DRM protected object.

It should be noted that it would be good to have a solution for download that is not dependent on OMA DRM. To be able to use download DRM it is required to distribute the Rights Objects; this calls for an infrastructure to support the delivery. Furthermore, Rights Object delivery in OMA

DRM is not designed with multicast environments in mind, and it is not obvious that it would work as currently specified.

Hence, from now on we will assume that the protection is focused on the channel and not of the actual object being delivered.

3 Double Protection Issues

As stated in the introduction, the MBMS delivery channel should not consider the semantics of the object to be delivered except for the case where the object is already protected, e.g., it is a DRM protected object. The reason for this is that double encryption is unnecessary overhead. However, it might still be a good idea to apply Source Origin Authentication (SOA) protection (see [6]) on the MBMS channel as the following example shows. Assume that an ME downloads a large DRM object, and that the packets on the MBMS channel are not SOA protected. In that case, if an attacker sends forged packets that fit the download stream (it requires some skill from the attacker to construct and insert valid FLUTE packets, but it is doable), the ME will not notice this. When the ME has downloaded the entire DRM object and tries to verify the integrity (if integrity protection is applied), it will fail due to the forged packets (easy to perform DoS attacks). Note that simple integrity protection under the group key will not suffice, but SOA is required [6] (this is also acknowledged in [2], where TESLA [9] is recommended).

To avoid double encryption, the BM-SC should have knowledge about which objects are already protected. How this knowledge is conveyed to the MB-SC from the content provider is FFS. There are some possibilities, e.g., the BM-SC could check the object to be delivered to determine if it is protected, or the content provider could indicate out of band that an object is protected.

When the BM-SC is to deliver a protected object, the security protocol can be disabled or NULL-transforms can be used. NULL-transforms exists for most security protocols, e.g., DTLS and SRTP.

4 Protection Schemes

The MBMS delivery channel can be protected using several existing protocols. In [4] it was shown that the security protocol used should be at the transport/application layer and must be able to run over unreliable transport; this requirement rules out several candidates, e.g., IPsec and TLS.

4.1 S/MIME

The use of FLUTE as a mechanism to achieve reliability of the transport hints at the option to use S/MIME [5] to protect the object. Since the type of the object is specified by a MIME type in FLUTE this may at first sound like a promising approach. S/MIME creates a secure container in which the object is put. This container may be secured entirely using symmetric key cryptography, based on pre-shared keys. In addition to this S/MIME provides strong cryptographic algorithms, e.g., AES and HMAC/SHA-1.

Different objects can be delivered in one and the same FLUTE session. S/MIME as a protection mechanism fits well into this model. Each object would be an S/MIME protected object. The objects could be protected under different or the same MTK. However, all objects delivered in the same FLUTE session would be protected by the same MSK. If it is a requirement to be able to use

several MSK:s in the same FLUTE session, some way of signaling which objects are associated with which MSK must be introduced.
S/MIME would fit neatly into the two-level key scheme proposed in SA3. There would however be a difference from the streaming case in how the keys would be delivered. In the streaming case, the MSK is delivered point-to-point, and MTK is proposed (see [10]) to be delivered over the multicast channel in a separate message. If S/MIME is used, the MSK is delivered in the same way as for streaming. The difference is in how the MTK is delivered; since S/MIME is a "self contained" container, it contains the payload encryption/authentication keys in the container itself (they can be encrypted with a pre-shared key). Figure 1 depicts a logical view of an S/MIME container, as it could be used for MBMS download. In the figure, $E_k(x)$ denotes encryption of object x under key k, and $MAC_k(x)$ denotes authentication tag computed over object x using key k.

Figure 1. Logical view of how S/MIME could be used in MBMS download.

If one whishes to have SOA together with S/MIME, there is no way of achieving this using symmetric key cryptography. There is however possible to sign the container, but this requires the use of public key cryptography. A signature would add around 1 kB per object in overhead.

S/MIME would be very easy on the bandwidth, since it only would add approximately 400 bytes per *object.* This in contrast to packet protecting schemes that add overhead on a *per packet* basis.

The main drawback of S/MIME is that it provides object protection, and not channel protection. This implies that the integrity of the object is not verified until the entire object is downloaded. Assuming that an attacker manages to insert a fraudulent packet in the FLUTE stream (see Section 3) the integrity check of the object will fail; this gives the opportunity for a DoS attack.

To mitigate this, one solution could be to partition the object into several smaller objects, and apply S/MIME on each and every one of them. This will lead to overhead far worse than if dedicated channel protection schemes are used. Furthermore, there is no standardized way of achieving this subdivision of the objects, so additional standardization has to be done.

Since S/MIME does not protect the FLUTE packets, there is also the issue of the risk of leaving that in the clear. The implications of leaving, e.g., the File Description Table, unprotected is FFS.

4.2 DTLS

One problem with the TLS protocol is that it requires that the underlying transport is reliable (in practice this means TCP). To overcome this shortcoming, a new protocol, Datagram TLS [1] has been proposed to the IETF. DTLS tries to re-use as much as possible from the TLS protocol, and basically just adds reliability mechanisms to accommodate for the possibility of lost and re-ordered packets. This is accomplished by adding sequence numbers to the records, and acknowledgement messages to the handshake process.

Since DTLS is derived from TLS, it has inherited the built in handshake and point-to-point design philosophy. Because of this DTLS cannot be used as is in a multicast scenario. To deal with the unfriendliness to multicast modifications to DTLS will be required. However, it is very difficult to drive changes to the TLS protocol in the IETF since it is so well established. Furthermore, TLS requires the use of public key cryptography, which might not be desirable in all cases, e.g., when thin clients are involved.

Despite the fact that IETF frown upon severe modifications to TLS there is a proposal to use symmetric key cryptography in TLS [3]. This draft does however, still mandate the four-message handshake, i.e., the key management is tightly coupled to the data protection protocol and is still not multicast friendly.

Since DTLS is designed for point-to-point links, the integrity protection relies on the shared key (there is no concept of a group key for DTLS). That is, it is not possible to obtain SOA of data.

DTLS is based on a TLS, and hence uses the same idea of a record-layer. The record-layer adds approximately 30 bytes of overhead per record (in form of authentication tags etc).

To be able to incorporate DTLS in MBMS two tasks needs to be completed; the first being to see to that DTLS goes through the IETF standardization process, and the second (and probably more difficult and time consuming) is to decouple the key management from DTLS and provide it with a multicast friendly key management. Even though DTLS have many nice properties, it is not foreseen that it can be used due to the problems described above.

4.3 SRTP

SRTP [7] was initially designed for protection of real-time data transported over RTP, and not for download in general. However, the streaming nature of FLUTE makes the use of SRTP for protection in the MBMS download case quite natural. In contrast to DTLS, SRTP already has RFC status in IETF, and is designed to protect RTP, which work with multicast and groups as well as with point-to-point connections. Furthermore, work is ongoing [8] in IETF to standardize SOA for SRTP. It is possible to use SRTP both in the streaming and the download case; the benefit of re-usable functionality is obvious.

Since SRTP does not include key management, it requires a companion protocol. In the case of MBMS this is an advantage, since it can be integrated with GBA and the choice of key management protocol can be based on the MBMS model instead of having to modify the model to fit a particular key management protocol. The MKI field in SRTP can be used to signal which MTK is used for protection of the data in the given packet.

To enable the use of SRTP, there is some work to be done. In particular, there is a need to specify the usage of FLUTE over RTP (see Figure 2).

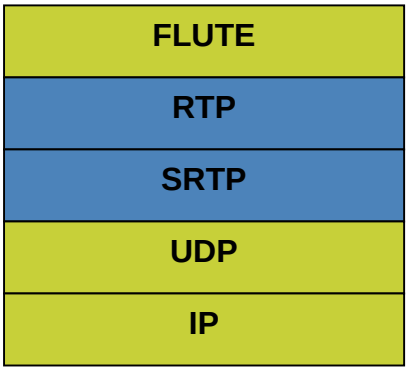

Figure 2. The IP-stack with the dummy RTP layered added to enable SRTP encryption. The blue layers would be inserted for the sake of security.

The fields in the RTP header that needs to be defined for SRTP to function correctly, is the SSRC field and the SEQ field. SRTP requires the SSRC field to be unique inside the RTP session; this is not really a problem in the MBMS setting, since the only sending party is the BM-SC. Hence the SSRC can be picked at random as specified in RTP [RFC3550]. When it comes to the SEQ field, SRTP uses it for synchronization of the cipher and also for key refresh rate (although key refresh is probably not an issue for MBMS). If the RTP-sender, i.e., the BM-SC, increases the SEQ by one (mod 2^{16}) per packet, this is enough to allow SRTP to keep the synchronization.
All other fields should be set to zero (with exception of the version field, which shall be set to 2, and possibly the payload type field).

The overhead added by SRTP would be a variable length MAC, and possibly a variable length MKI per packet. This would typically sum up to around 24 bytes per packet.

Previous discussions concluded that in order to apply SRTP to MBMS traffic, it is required to define that the payload in the RTP packets is built from RMT building blocks. The signaling would be carried in an SDP description. To signal that the RTP stream is carrying RMT building blocks, one solution is to register a new payload type. New payload types are assigned by specifying a new RTP profile. An RTP profile is a complex machinery, and will take long time if driven through the IETF. Hence, this is not the preferred way to go. If a FLUTE profile was available, the SDP description would contain an m-line along the following:

m=application port RTP/SFLT/UDP,

where application signals that the content is secure flute packets, port is the port to receive packets on, RTP/SFLT is the new RTP profile.

Alternatively, the audio/video profile (AVP) can be reused. The modification required in this case is that a new MIME-type has to be registered. The SDP description would then contain an m-line looking something like this:

m=application port RTP/AVP mime,

where mime would be the newly registered MIME-type of FLUTE.

4.4 Comparison Table

This section summarizes the properties of the considered solutions. Note that the row which describes overhead is a bit special in the sense that all three protocols have a different metric. That is, for S/MIME there is only one authentication tag per object (i.e., file) that is downloaded. DTLS adds overhead on its record-layer; a record could potentially span across several UDP packets. SRTP adds overhead on a per packet basis. This implies that the overhead added by S/MIME is far less than any of the other two. DTLS and SRTP add more or less the same.

Note that for S/MIME there might be no need for further standardization (i.e., if it is acceptable to use only one object).

The row on re-usability refers to the possibility to share the component between the streaming case and the download case.

	S/MIME	**DTLS**	**SRTP**
Standardization Work required	Possibly object partitioning (not ready before R6)	DTLS, SOA (not ready before R6)	RTP profile/MIME-registration, SOA (possibly ready for R6)
Multicast friendly	Yes	No	Yes
SOA protection	Signatures	No	TELSA (ongoing)
Protocol stack suitability	Good	Good	Dummy RTP layer
Coupled key mgmt.	No	Yes	No
Overhead	Small (per object)	Small (per record)	Smaller (per packet)
Streaming Re-usability	No	Yes	Yes

5 Conclusions

As Section 4 shows, the scheme that introduces least problems for security of data download in MBMS is S/MIME. The main limitation of S/MIME in this context would be that of the possibility of a DoS attack. If the risk for such an attack is deemed acceptable, S/MIME is suitable. On the downside, S/MIME is not suitable for the streaming case so re-use with respect to streaming will not be possible.

It should be noted that it is very difficult to find a mechanism that works both for streaming and download, and fits the requirements. SRTP could possibly be used with some additional work.

6 Proposal

We propose to use S/MIME for protection of MBMS download traffic. S/MIME also fits well in the two-level key hierarchy proposed in [10].

7 References

[1] Rescorla and Modadugu, ``draft-rescorla-dtls-00´´, IETF draft, 2004, work in progress
[2] Paila et. al., ``draft-ietf-rmt-flute-07´´, IETF draft, IETF draft, 2003, work in progress
[3] Gutman, ``draft-ietf-tls-sharedkeys-02´´, IETF draft, 2003, work in progress
[4] Ericsson, MBMS - Security layer selection, S3-020533, October 2002
[5] Ramsdell et. al. ``S/MIME Version 3 Message Specification´´, IETF RFC2633, 1999
[6] Ericsson ``On the need for integrity and source origin authentication in MBMS'', April 2004
[7] Baugher, et.al., ``Secure Real-time Transport Protocol (SRTP)´´, IETF RFC3711, 2004
[8] Baugher and Carrara, ``draft-ietf-msec-srtp-tesla-00´´, IETF draft, 2004, work in progress
[9] Canneti et. al., ``draft-ietf-msec-tesla-spec-00´´, 2002, work in progress
[10] Nokia, ``Further updates on Combined model for MBMS security'', February 2004