3GPP TSG SA WG3 Security — S3#35 **S3-040809**
5 - 8 October 2004
St Paul's Bay, Malta

Title: **Updated: MBMS Download Protection using XML**

Source: **Ericsson**

Document for: **Discussion and decision**

Agenda Item:

Work Item: **MBMS**

1. Introduction

A contribution on XML protection was presented in S3 S4 joint meeting [1]. This contribution is an update of that contribution. An accompanying CR will implement the changes in TS 33.246 [4].

2. XML Signatures

XML signatures are defined in RFC 3275 [2]. The usage of the word *Signature* in [2] is not the same as the common usage of the word in that it also encompasses message authentication codes (MAC:s). In the following we will keep the terminology of [2], but we will only consider the symmetric key based MAC:s.

2.1 SignatureMethod

As mentioned above, it is possible to use a MAC instead of a public key based signature. RFC 3275 [2] defines the use of HMAC/SHA-1 for creating the MAC. The output of the algorithm can be truncated, but there is no apparent reason why it should be in this case.

Example:

```
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1">
<HMACOutputLength>160</HMACOutputLength>
</SignatureMethod>
```

The HMACOutputLength tag can be omitted since the default is 160 if it is left out.

The HMAC/SHA-1 is computed over the SignedInfo element. That is, everything inside the SignedInfo start and stop tags is covered (see Section 2.5).

2.2 DigestMethod

To produce a signature, a digest of what is to be signed is produced. The digest is then Base64-encoded and put into a DigestValue element. This DigestValue element is finaly signed by performing a MAC computation which includes the DigestValue element.

Example:

```
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
```

2.3 KeyInfo

The KeyInfo element tells the receiver which key (or how to retrieve the correct key) that was used when signing. In the case of MBMS this element can be used to signal key ID:s of the MBMS Service Key (MSK) and MBMS Traffic Key (MTK) used to protect the given file. Of course the MTK is not used as input key to HMAC/SHA-1 directly, but instead the integrity key MTK_I derived from it (the obvious one-to-one mapping is used to retrieve the correct MTK_I). Note that how the MTK_I is derived from the MTK is out of scope for this document.

Example:

```
<KeyInfo>
    <KeyName>MSK_ID || MTK_ID</KeyName>
</KeyInfo>
```

In the example, *MSK_ID* || *MTK_ID* is the concatenation of the identities of the respective keys.

2.4 Reference (Signing both the FDT and the file(s))

The flexibility of XML signatures provides the possibility to reference objects to be included in the signature that is not part of the XML document per se, i.e., for MBMS it is possible for the same signature to cover both the FDT and the files. The obvious way would then be to envelope the signature around the FDT, which is already described in XML. The signature would carry a reference to the actual files that are to be signed. The reason for that there may be several files is that there might be for instance a video and an audio file that are delivered as separate entities. An example of the usage of references is given in Section 2.6.

2.5 Putting the pieces together

The following example shows how the elements of the previous sections fit together to integrity protect the downloaded file and the corresponding FDT. Please note that the actual values of digests, MAC:s etc. are dummies, and *not* correct in the sense that they do not correspond to what is achieved by applying the specified algorithm to the actual data. Assume that the FDT for the file located by the URI "www.example.com/a_file" is given by:

```
<?xml version="1.0" ?>
<FDT-Instance Expires="3285666382">
    <File TOI="1"
    Content-Location="www.example.com/a_file"
    Content-Length="679936"
    Content-MD5="Tt0dxyJfU9mX9YubHsoYUA==" />
</FDT-Instance>
```

This XML description of the FDT is given an enveloping signature according to RFC 3275 [2] to generate the following:

```
<?xml version="1.0" ?>
<Signature Id=”a_file.sign”
xmlns="http://www.w3.org/2000/09/xmldsig#">

<SignedInfo>
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1">
        <HMACOutputLength>160</HMACOutputLength>
        </SignatureMethod>

        <Reference URI="#FDT"/>
                <DigestMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <DigestValue>hf2UIfdo4uihHIJoOj3PjdIOPjdowF=</DigestValue>
        </Reference>

        <Reference URI="www.example.com/a_file"/>
                <DigestMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
        </Reference>

</SignedInfo>

<SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>

<KeyInfo>
        <KeyName>MSK_ID || MTK_ID</KeyName>
</KeyInfo>

<dsig:Object xmlns="" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Id="FDT">
        <FDT-Instance Expires="3285666382">
                <File TOI="1"
                Content-Location="www.example.com/a_file"
                Content-Length="679936"
                Content-MD5="Tt0dxyJfU9mX9YubHsoYUA==" />
        </FDT-Instance>
</dsig:Object>
</Signature>
```

As can be seen above, the original FDT description is enveloped in a signature-description using the reference/object method. If more than one file is corresponding to the given FDT, there would be a reference element for each of the files. Note also that it is possible to sign only the FDT and leave the file unprotected (simply by not referencing it with the Reference element).

One problem with the above approach of having the same signature covering both the FDT and the actual files is that if an attacker forges the FDT, the receiver would have to download the entire file before the validity of the signature could be verified (and the download deemed invalid). To remedy this, two Signature elements can be used in the FDT XML-description:

- The first Signature element would look like in the example above, but with the Reference element pointing to URI www.example.com/a_file taken away (it is actually moved to the second signature element). Hence it only contains one Reference element to the local Object (which contains the FDT).
- The second Signature element would look exactly the same as the previous one, except that it contains a Reference element pointing to URI www.example.com/a_file instead of to the local Object.

Note that the only difference between the two signature elements is the Reference elements.

3. XML Encryption

As pointed out in S3-040557 [1], encryption of parts of the FDT can be performed using XML encryption [3] (although the necessity of confidentiality protection of the FDT is questionable). Of higher importance is of course the encryption of the actual files that are downloaded. This can be accomplished by use of the CipherReference element. Note that XML encryption reuses elements from XML signatures, e.g., the KeyInfo and KeyName elements.

The following examples will show how the encryption is specified using XML. The UE is then expected to decrypt the downloaded files before continuing with the normal processing of the file. Note that the XML descriptions below only specifies how a certain object is confidentiality protected; the actual processing steps and behavior of the UE (i.e., the semantics) is described in [3].

```
<?xml version="1.0" ?>
<FDT-Instance Expires="3285666382">
        <File TOI="1"
        Content-Location="www.example.com/a_file"
        Content-Length="679936"
        Content-MD5="Tt0dxyJfU9mX9YubHsoYUA==" />
</FDT-Instance>
```

The above example in encrypted form could look like the code below if also the actual file is encrypted:

```
<?xml version="1.0" ?>
<EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
xmlns="http://www.w3.org/2001/04/xmlenc#">

        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:KeyName>MSK_ID || MTK_ID</ds:KeyName>
        </ds:KeyInfo>

        <EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>

        <CipherData>
                <CipherValue>A523F3478…</CipherValue>
        </CipherData>
```

```
</EncryptedData>

<EncryptedData
xmlns="http://www.w3.org/2001/04/xmlenc#">

        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:KeyName>MSK_ID || MTK_ID</ds:KeyName>
        </ds:KeyInfo>

        <EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>

        <CipherReference URI="http://www.example.com/a_file"></CipherReference>
</EncryptedData>
```

The first EncryptedData element is the encryption of the original FDT element, whereas the second and last EncryptedData element is the specification of how to decrypt the actual file (note that the document does not contain the encrypted file, but merely references the file using the same URI as in the FDT). The keys required to decrypt the message (and the file) can be retrieved with the aid of the KeyInfo element (which is reused from the xmldsig name space).

Note that the prime reason (in [1]) for confidentiality protecting the FDT was privacy concerns, i.e., the location (and name) of the actual file downloaded should be hidden. Since the CipherReference element needs to include this information there are some possibilities how to solve the privacy issue. Firstly, it is possible to put the last EncryptData element (which contains the CipherReference) into another EncryptedData element (in the same way the FDT was encrypted). This approach costs another encryption operation (of a small object though), but achieves high security. Another idea is to use a URI that cannot be associated with the content of the file. This approach requires less processing, at the cost of that it reveals what file is downloaded to anyone who has previously downloaded the file (and hence knows the URI).

4. Encryption and integrity protection combined

Now, it is only a matter of combining the two. Since it is good practice to first encrypt and then apply integrity protection, that is what we suggest. This means that we first apply XML encryption as described in Section 3. After that the document that is output from that process is signed using the method in Section 2. Note that there are no additional XML schema definitions required.

5. Conclusion and proposal

The paper has shown, how XML encryption and XML signatures can be used to protect MBMS download. The protection is achieved through purely symmetric key methods.

It is proposed that this protection scheme is used for protecting MBMS download. Accompanying CR will implement the changes in TS 33.246.

6 References

[1] TD S3S4J0003, MBMS Download Protection using XML
[2] Eastlake et al, “XML-Signature Syntax and Processing”, RFC 3275, IETF
[3] Eastlake et al, “XML Encryption Syntax and Processing”, W3C
[4] TS 33.246, MBMS Security