3GPP TSG SA WG3 Security — S3#36 **S3-040902**

23 - 26 November 2004

Shenzhen, China

Title:	**Overhead and Performance Comparison of OMA DRM V2.0 DCF and XML for MBMS Download Protection**
Source:	**Nokia**
Document for:	**Discussion**
Agenda Item:	**MBMS**
Work Item:	

1. Introduction

In [1], we proposed an extension of OMA DRM V2.0 DCF [2] for MBMS Download Protection. In an accompanying discussion paper [3], we provide an update to the initial proposal with details of how integrity protection can be achieved and how the FLUTE FDT (File Description Table) can be protected.

In this paper, we provide a comparison study of using the proposed DCF and XML for MBMS download protection in terms of overhead required as well as performance. We will show that as a simple binary format, DCF incurs much less overheads than XML. It is also expected that performance for processing binary DCF objects should be better than XML. We therefore believe that DCF should be selected for MBMS download protection.

2. Overhead Comparison

For use by MBMS download protection, both DCF and XML add a certain amount of overheads per downloaded content (rather than per packet). This section estimates and compares the amount of overheads needed in each case.

2.1. DCF Overheads

The DCF format is structured around an object-oriented design of boxes. A basic box has two mandatory filed, size and type. The basic box therefore requires a total of 8 bytes overhead, as shown in Table 1.

Name	Type	Size (bytes)
Size	unsigned int(32)	4
Type	unsigned int(32)	4

Table 1 Basic Box

If size of a basic box is of the type “largesize”, the size field will be set to 1, and an extra field “largesize” of type unsigned int(64) will be defined to represent the actual size. In which case, 8 bytes more are required, resulting in a total of 16 bytes.

A Fullbox extends the class of basic box, and can be represented as having the following mandatory fields:

Name	Type	Size (bytes)
Size	unsigned int(32)	4
Type	unsigned int(32)	4
Version	unsigned int(8)	1
Flags	unsigned int(24)	3

Table 2 FullBox.

The following table summarizes the overheads needed in each type of box:

Box Type	Size (bytes)
Basic Box	8
Basic Box, Large size	16
FullBox	12
FullBox, Large size	20

According to the different types of box in a DCF, as well as the various fields in each of these boxes, we can compute the amount of overheads as follows:

Field	Nesting Level	Description	Size (bytes)
Fixed DCF File header	0	‘odcf’	4
Fixed DCF File version	0	0x00020000	4
OMA DRM Container Box	0	FullBox, Large Size	20
DCF Headers Box	1	FullBox + various fields	13 + ContentTypeLength
Common Headers Box	2	FullBox + various fields	40 + Length(Key_id)
Content Object Box	1	FullBox + various fields	20
		Subtotal (for Encryption only)	101 + ContentTypeLength + Length(Key_id)
FreeSpaceBox	0	Box	8
MBMSSignature Box	1	FullBox + various fields	33
		Subtotal (With Signature)	41
		Total	142 + ContentTypeLength + Length(Key_id)

For simplicity, assume both ContentTypeLength and Length(Key_id) equal 20 bytes. We arrive at the following estimation shown in Table 3. If FDT protection is desired, two times the overhead will be needed.

Overhead incurred (bytes)	Content only		Content + FDT	
	Encryption only	Encryption and signature	Encryption only	Encryption and signature
DCF	141	182	282	364

Table 3 DCF Overhead.

2.2. XML Overheads

The overheads incurred by using XML security are estimated based on the examples below. In each case, an example XML document is given, and the overhead is estimated by counting the number of characters needed. It should be noted that there are multiple ways of representing data using XML, but we believe the estimation based on the following examples would be representative.

In each case, the MSK_ID || MTK_ID field is assumed to be 20 bytes, just as in the DCF case.

2.2.1. Content only protection

2.2.1.1. Encryption only

```
<?xml version="1.0" ?>

<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"/>

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

The above XML document comprises of 354 characters (Note that newlines and carriage returns are not counted.)

2.2.1.2. Encryption and Signature

```
<?xml version="1.0" ?>

<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"/>

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

<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">
     </SignatureMethod>
```

```
      <Reference URI="">

      <Transforms>
      <Tranform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
      </Transforms>

      </Reference>

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

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

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

</Signature>
```

The above document comprises of about 937 characters.

2.2.2. Both Content and FDT protection

2.2.2.1. Encryption only

```
<?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 CipherValue of the FDT are not counted, as it is considered to be the content itself. Note however that the encrypted FDT has to be base64 encoded, which increases its size by about 33%. The number of characters in the above document is counted to be 715.

2.2.2.2. Encryption and Signature

```
<?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>

<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">
      </SignatureMethod>

      <Reference URI="">

      <Transforms>
      <Tranform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
      </Transforms>
```

```
      </Reference>

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

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

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

</Signature>
```

The above document is estimated to comprise of 1298 characters, not including the encrypted FDT itself.

2.3. Comparison

Putting the numbers together, we obtain the following table. Due to the fact that DCF is a simple binary wrapping format, whereas XML is text-based, XML incurs about 2.5 to more than 5 times the overhead compared to DCF.

Overhead incurred (bytes)	Content only		Content + FDT	
	Encryption only	Encryption and signature	Encryption only	Encryption and signature
DCF	141	182	282	364
XML	354	937	715	1298
DCF:XML	1:2.5	1:5.1	1:2.5	1:3.6

Table 4 Overhead Comparison, DCF vs. XML.

Table 5 tabulates the percentage overhead for using DCF and XML for typical downloaded contents. For contents of around 10k bytes, DCF overhead is around less than 2% (for content only protection) while XML adds nearly 10% of overhead. It is true that as file size of downloaded content increases (e.g. up to 1Mbytes), overheads incurred by both DCF and XML will be negligible. However, we believe that most mobile downloaded contents would tend to be small in size. And also, full-length videos will likely be streamed to the mobile device, rather than using download mechanism.

Content Type	Typical File size	Content Only		Content + FDT	
		DCF Overhead	XML Overhead	DCF Overhead	XML Overhead
Java games	10k – 100k	0.18% - 1.82%	0.94% - 9.37%	0.36% - 3.64%	1.30% - 12.98%
jpg images (640x480, True colors)	10k – 40k	0.46% - 1.82%	2.34% - 9.37%	0.91% - 3.64%	3.25% - 12.98%
Midi Ringtones	3k – 10k	1.82% - 6.07%	9.37% - 31.23%	3.64% - 12.13%	12.98% - 43.27%
Short video clips (e.g. video/H264-2000, resolution 128x96, 10 sec duration)	~ 100k	~ 0.18%	~ 0.94%	~ 0.36%	~ 1.30%
Short sound clips (10 sec)	~ 10k	~ 1.82%	~ 9.37%	~ 3.64%	~ 12.98%
Mp3 (128kbps, 5 mins)	~ 5M	negligible	negligible	negligible	negligible

Table 5 Percentage overheads for typical content types.

3. Performance Comparison

There is no direct performance comparison conducted between DCF and XML. However, the baseline is that DCF is a simple binary format, which is fairly easy to parse, while XML is text-based and requires more processing (including canonicalization). There are some studies in the literatures, which compare XML and other schemes in providing security services such as digital signatures. The results provide some insights to the performance issues of using XML versus other binary formats. These are briefly discussed as follows.

A recent study compares XML and ASN.1 for performance in providing signed messages [4]. ASN.1 is a description language for abstract data. There are multiple encoding rules defined that convert an ASN.1 object to data stream that is transmitted on wire. The Binary Encoding Rule (BER) condenses the ASN.1 representation to a binary data stream. Therefore by looking at the comparison between ASN.1/BER and XML-based schemes, we could get a feel of how binary formats, such as DCF, will perform compared to XML. The result shows that XML signed document is about 13 to 20 times the size of a corresponding ASN.1 signed object, depending on the complexity of the object to be signed. In terms of the processing times needed for signature verification, the result shows that ASN.1 outperforms XML by 450 percent for a simple object, and close to 1000 percent for complex object.

Another performance study compares transport level security (SSL) and various XML-based security mechanisms for Grid Services [5]. In this study, there is an interesting break down of the processing times needed in various processing phases of XML-Signature creation and verification. In verification, the various processing phases includes conversion to DOM (Document Object Model), certificate path validation, canonicalization, signature verification, and other operations. The result shows that canonicalization takes up most of the time in the verification process. For

example, in one of the test, 1395.5 msec is used in canonicalization out of 1445.8 msec of total processing time. Such processing overhead is not required if the data is represented in binary format.

There has also been discussion within the Wapforum regarding the performances of XML Digital Signatures compared to some other schemes, such as S/MIME. The conclusions are in general in line with the discussions above. In terms of processing times, there are test results that show XML requires a lot more processing time for creating and verifying digital signatures, compared to S/MIME.

These are just some of the examples from the literatures on XML performance. It shows that typically, binary format (S/MIME, ASN.1) outperforms text-based XML in terms of digital signature creation and more importantly, signature verification.

4. Conclusions

In this discussion paper, we compare DCF and XML for MBMS download protection in two aspects, namely, the file size overhead, and performance in processing.

In terms of file size overhead, we estimated that XML adds 2.5 to 5.1 times more overhead than DCF, due to its text-based nature. For mobile contents that are typically small in size, this comparatively large overhead when using XML could be significant, and takes up more precious over-the-air bandwidth. In terms of processing, there are performance studies in the literature that indicates the complexity in processing XML signatures, due to the fact that XML signatures are text-based, and that extra time-consuming processing such as canonicalization is needed. These studies show that if performance is major concern, binary formats are more preferable.

XML security has its own advantages. Being text-based, XML documents are easy to understand and manipulate by human. Consequently, application development may be easier. Besides, XML-encryption and XML-signature also have the flexibility to encrypt and sign portions of an XML document. For the case of MBMS download protection, however, these advantages are not so important. The most crucial considerations should be the file size overhead, which increases usage of the precious over-the-air bandwidth; and also the processing requirements at the mobile terminals, as they are limited in terms of processing power and other resources (battery, memory, etc). Therefore, a binary format such as the simple OMA DRM V2.0 DCF should be a better choice than XML for protecting MBMS downloaded contents.

5. References

[1] S3-040781 Extensions to OMA DRM V2.0 DCF for MBMS Download Protection, S3#35, Oct 2004, Nokia.

[2] DRM Content Format, OMA-DRM-DCF-v2_0-20040715-C, www.openmobilealliance.org.

[3] S3-040xxx An Update to Using OMA DRM V2.0 DCF for MBMS Download Protection, S3#36, Nokia.

[4] D. Mundy and D. Chadwick, “An XML Alternative for Performance and Security: ASN.1”, IT Professionals, IEEE Computer Society, Jan/Feb 2004, pp. 30-36.

[5] S. Shirasuna, A. Slominski, L. Fang, and D. Gannon, “Performance Comparison of Security Mechanisms for Grid Services”, 5th IEEE/ACM International Workshop on Grid Computing, Nov 8th 2004, Pittsburgh, USA.