Secure Architecture for the Digital Rights Management of the M-Content

ION IVAN, CRISTIAN TOMA, MARIUS POPA, CĂTĂLIN BOJA
Department of Economic Informatics
Academy of Economic Studies – Bucharest
Romana Square, No. 6, Bucharest
ROMANIA
ionivan@ase.ro cristian.toma@ie.ase.ro marius.popa@ase.ro catalin.boja@ie.ase.ro

Abstract: The new trend in mobile applications world is to securely ensure a variety of content for all the customers. Since the providers invest large amounts of money in producing multimedia content such as m-applications, presentations, advertising, music clips and games, there should be considered some sensitive problems regarding the "forward lock" issue, in particular the content transfer from one mobile device to another. This paper presents models for handling digital rights management problems and a proposal for a practical distributed secure architecture used in a complex model for massive secure distribution of m-contents. Also, the concepts related to m-application, xml signature, keys management, and secure architectures are combined in a practical manner that helps the researchers, designers and developers from IT&C field to consider multifarious approaches of the digital rights management issues.

Key-Words: digital rights management, m-application, secure architecture, keys management

1 Introduction

This chapter specifies the terms that will be used in the paper and the basic concepts of the m-applications, digital rights management and secure architectures.

A *mobile application – m-application* – is a byte code that is running on a mobile device and is using the internal memory, microprocessor, and/or SIM – Subscriber Identity Module – features.

The providers, which collaborate with mobile operators and vendors, supply the mobile content – *m-content* – such as m-application (financial, office apps and games), movies and music clips, presentations, and pictures. The m-application is running on an infrastructure formed by: mobile devices, standards and communications protocols and processes.

There are three main methods of developing mobile-wireless applications:

- *M-applications WAP* (using the phone's WAP browser – an sample of WAP m-application is presented in e3com [9]) – the phone has a client WAP browser that parses and interprets the information from different markup languages like: WML or XHTML, in such way like Netscape Communicator or Internet Explorer "understand" HTML. The possibility for m-computing (calculus for complex algorithms like crypto-graphics one) is very small.
- *Java 2 Micro Edition Applications* – J2ME – some phones implement KVM – Kilo Virtual Machine ("KVM's big brother is JVM – Java Virtual Machine"), so the procedure for building the programs is more like building programs for PC's – personal computers. The application *eBroker* is developed in this way [8] – it ensures the financial transaction performed by the mobile device in a secure manner. The m-applications are running within the device's microprocessor and use a special part of the device's memory. The company Sun Microsystems provides APIs J2ME for developing MIDP – Mobile Interface Device Profile applications (midlets).
- *Open Operating Systems Applications* – for mobile phones and devices that have their own operating system – Symbian OS, Linux or Microsoft Mobile OS, the client side of m-applications is built using SDKs (Java, C++ or OPL-BREW) provided by the producers of operating systems and/or devices.

There were possibilities of sending the m-content in a secure manner using the last two approaches of them-application development, but it is quite difficult to "pre-install" software on all phones in a mandatory manner, so a new technology such as Digital Rights Management took their place.

Digital Right Management – DRM is a set of specifications that designates a variety of standards for certain particularities and features of the m-content. The mobile device software, content providers and mobile operators control the usage of the downloaded

media objects. Trivially speaking, the digital rights management ensures the provider of the content that once a mobile device downloads the multimedia application or clip, it is impossible to forward the clip to another device – there is no "Send" option.

The development within communication technology field allows securing the multimedia m-content by using and developing *secure architectures* that practically implement conceptual informatics security models and patterns.

2 Digital Rights Management Models

The DRM – Digital Rights Management Models for m-content's delivery is a key concept presented in fig.1:

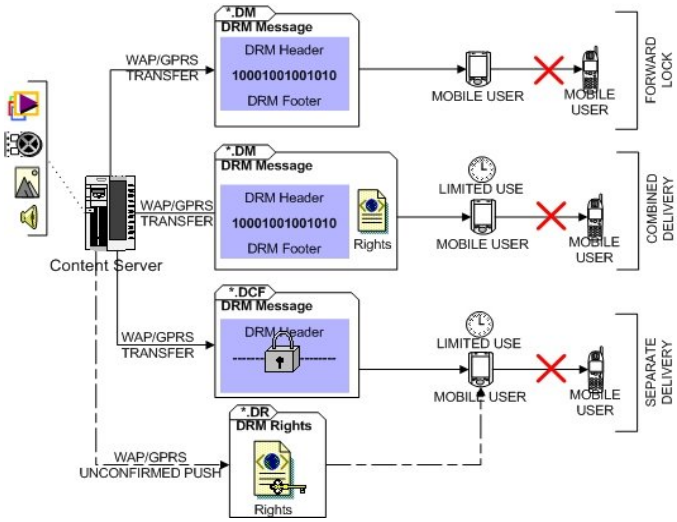

Fig. 1 DRM models for content delivery

According to [3] and represented in figure 1, there are three DRM models for the content delivery:

- *Forward-lock* – the content provider sends to the mobile browser a binary file (image, movie, game or application) with special header and footer like in table 1 with "*dm*" extension. The mobile browser launches an application called the DRM agent that allows the browser to display and play the m-content without a "*Send*" option, so the end-user has no possibility of forwarding the content to another device via Bluetooth or MMS.

Table 1–Forward-lock Representation of "dm" file

--boundary-1
Content-type: image/jpeg
Content-Transfer-Encoding: binary

ÿØÿà…Binary representation of the M-CONTENT
--boundary-1--

- *Combined-delivery* – before the binary content there is an XML representation of the "rights object" like in table 2 (encapsulated also in a "dm" file), which allows the browser to play only 3 times between 01.10.2006 – 01.11.2006 and does not allow it to forward the m-content.

Table 2–Combined Delivery Representation of "dm" file

--boundary-1
Content-type: application/vnd.oma.drm.rights+xml
Content-Transfer-Encoding: binary

```
<o-ex:rights
 xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
 xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#/">
 <o-ex:context>
  <o-dd:version>1.0</o-dd:version>
 </o-ex:context>
 <o-ex:agreement>
  <o-ex:asset> <o-ex:context>
    <o-dd:uid>cid:http://content-id-here</o-dd:uid>
  </o-ex:context></o-ex:asset>
  <o-ex:permission>
   <o-dd:play>
    <o-ex:constraint>
     <o-dd:count>3</o-dd:count>
     <o-dd:datetime>
      <o-dd:start>2006-10-01T20:59:10</o-dd:start>
      <o-dd:end>2006-11-01T20:59:10</o-dd:end>
     </o-dd:datetime>
    </o-ex:constraint>
   </o-dd:play>
  </o-ex:permission>
 </o-ex:agreement>
</o-ex:rights>
```

--boundary-1
Content-type: image/jpeg
Content-ID: <http://content-id-here>
Content-Transfer-Encoding: binary

ÿØÿà…Binary representation of the M-CONTENT
--boundary-1--

- *Separate-delivery* – the model allows the content provider to send the m-content that is encrypted with a symmetric key as in table 3 and 4. Therefore, within the separate delivery model, the content provider first sends the binary encrypted data with a header, encapsulated as in table 3 and figure 1 in a "dcf" file. The browser of the mobile device requests or receives the "rights object" file (the XML encapsulated in a "dr" file) from the URL included in "Rights-Issuer" field from "dcf" file. The rights object, if not request, can be pushed using WAP (Wireless Application Protocol) MMS – Multimedia Message Service or Push message

(SI – Service Indicator or SL – Service Locator) mechanisms.

Table 3–Separated Delivery Representation of "dcf" file

```
□
□image/jpegcid:http://content-id-
here  gŽŒ┼Encryption-Method: AES128CBC
Content-Name: "NameOfContent"
Rights-Issuer: http://rights-issuer.com/content
Content-Description: "DescriptionOfContent"
Content-Vendor: "VendorName"
Icon-Uri: http://vendor.com/content-icon.gif
"¶{…Binary encrypt representation of the M-
CONTENT using AES-Rijndael symmetric key
algorithm in CBC mode
```

Table 4–Separated Delivery Representation of "dr" file

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE o-ex:rights PUBLIC "-//OMA//DTD
DRMREL 1.0//EN"
          "http://www.oma.org/dtd/dr">
<o-ex:rights
  xmlns:o-ex="http://odrl.net/1.1/ODRL-EX"
  xmlns:o-dd="http://odrl.net/1.1/ODRL-DD"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#/">
  <o-ex:context>
    <o-dd:version>1.0</o-dd:version>
  </o-ex:context>
  <o-ex:agreement>
    <o-ex:asset>
      <o-ex:context>
        <o-dd:uid>cid:http://content-id-here</o-dd:uid>
      </o-ex:context>
      <ds:KeyInfo>
        <ds:KeyValue>
          joVbFmkmi3bSO6gC98HE1Q==
        </ds:KeyValue>
      </ds:KeyInfo>
    </o-ex:asset>
  <o-ex:permission>
    <o-dd:play>
      <o-ex:constraint>
        <o-dd:count>2</o-dd:count>
        <o-dd:datetime>
          <o-dd:start>2006-09-27T20:59:10</o-dd:start>
          <o-dd:end>2007-09-27T20:59:10</o-dd:end>
        </o-dd:datetime>
      </o-ex:constraint>
    </o-dd:play>
  </o-ex:permission>
  </o-ex:agreement>
</o-ex:rights>
```

In conclusion, there are two ways of delivering the content rights object to the user, taking into consideration the number of files that are sent to the mobile device:

- to the consuming devices, together with media object (DRM Forward Lock and Combined Delivery Model);
- sending the rights separately from media content (DRM Separate Delivery Model).

Regardless of which of the three models is implemented a *download descriptor file* such as in table 5 can be used in order to improve the user experience.

Table 5–Download Descriptor Representation "dd" file

```
<media
xmlns="http://www.openmobilealliance.org/xmlns/dd">
  <DDVersion>1.0</DDVersion>
  <name>Name Of Product</name>
  <size>1234</size>
  <type>image/jpg</type>
  <vendor>Media Vendor Company</vendor>
  <description>Description</description>
  <objectURI>http://object-url</objectURI>
  <iconURI>http://icon-url</iconURI>
  <infoURL>http://info-url</infoURL>
  <nextURL>http://next-url</nextURL>
  <installNotifyURI>
    http://install-notify-url
  </installNotifyURI>
  <installParam>-param1 -param2</installParam>
</media>
```

The mobile device downloads the download descriptor file and the browser is redirected to the URL (the address between "<*objectURI*>" tag from "dd" file – table 5) that contains or generates the "dm" or "dcf" file depending on which of the DRM models present. The table 6 presents the MIME (Multipurpose Internet Mail Extensions) media types of the objects, according to the DRM message format.

Table 6–MIME media types

DRM method	MIME media types
Forward-lock	application/vnd.oma.drm.message
Combined delivery	application/vnd.oma.drm.message application/vnd.oma.drm.rights+xml
Separate delivery	application/vnd.oma.drm.rights+xml application/vnd.oma.drm.rights+wbxml application/vnd.oma.drm.content

The DRM message is based on a MIME multipart composite type in which one ore more objects are combined in a single body. The body of the DRM message must be according to the body of the multipart media type defined in RFC 2045 and 2046, chapter 5 [2]. The Digital Right Management message must contain one or two body parts, one for each object.

If HTTP (Hyper Text Transfer Protocol) or a MIME compliant protocol is used to transport the Digital Right Management message, the boundary delimiter

must be included as a parameter within the media type definition.

RFC 2045 defines a Content-Transfer-Encoding. This specifies how a body is encoded for transfer by some transfer protocol. A Content-Transfer-Encoding header must be included in the body part of the Digital Right Management message.

3 The DRM Secure Architecture Problem

In the real solutions that are implemented for important mobile operators such as Orange, Vodafone or T-Mobile, there are three types of platforms that implement the DRM model:

- *Platform for Microsoft Mobile OS smart phones* – mandatory use of the DRM Separated Delivery Model only in a "proprietary" manner. That means that it is necessary for the programs of the content providers to use Windows Media SDK, in order to generate undocumented headers for "wmv/wma" files. These headers include the link to where the rights object resides, that is, on a key management server produced by Microsoft only.
- *Open Platform* for all others mobile operating systems phones – Nokia, Sony-Ericsson, Motorola – dynamically build the "dd" and "dm" files (for the first two DRM models) or dynamically generate the "dd", "dcf" and "dr" files (for DRM Separate Delivery model) for all mobile devices that *do not* have Microsoft user agent;
- *Hybrid Platforms* – analyze the user agent of the mobile device and depending where it is Microsoft or some other, it dispatches to one of the previous platforms.

Since the development and the implementation of open documented architectures is strongly recommended, the proposed secure architecture for key management focuses on DRM Combined Delivery Model in Open Platform implementations.

The functional details of the analyzed model are highlighted in figure 2:

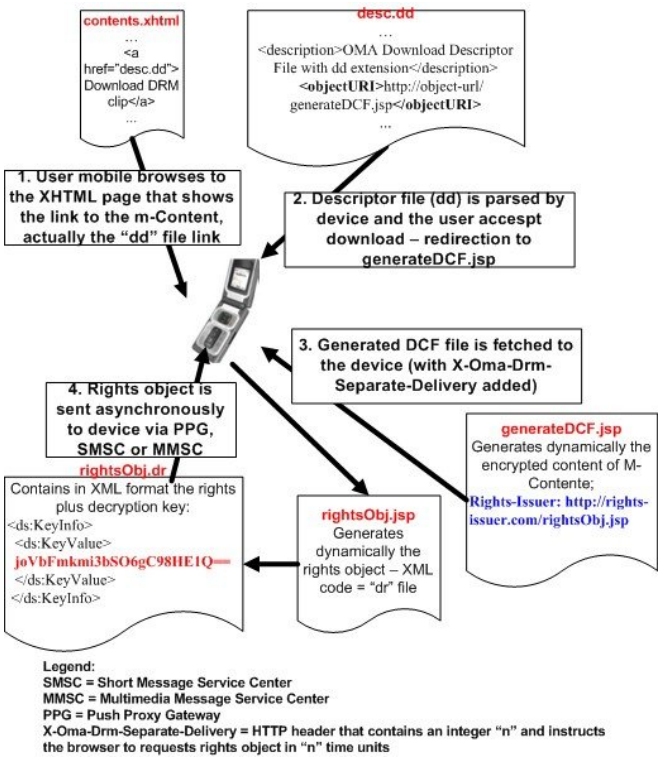

Fig. 2 Functional details of the DRM Separated Delivery Model implemented in *Open Platforms*

Figure 2 is self descriptive but it is important to point out that DRM Separated Delivery Model implemented in *Open Platforms* should generate three files: the download descriptor (.dd), the digital crypt m-content file (dcf) and digital rights object file (dr).

The super-distribution business model that is represented in figure 3, is one of our main focuses:

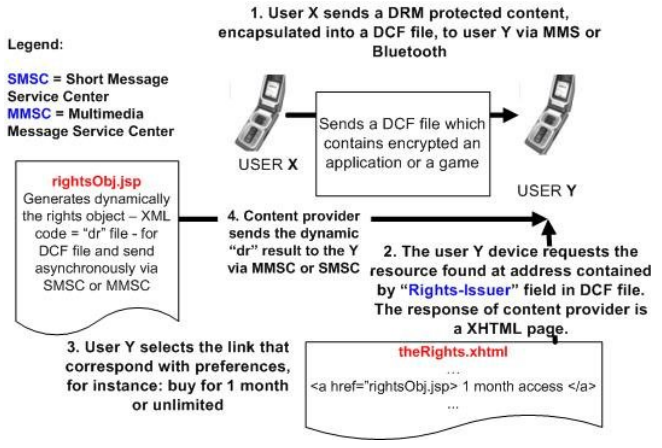

Fig. 3 Super-distribution business model

Flexibility is one of the features of the super-distribution case for a separate delivery. The sharing of

media objects without compromising any business model behind the rights is encouraged. For digital rights object delivery, the WAP push technology is used. The media object is passed from one mobile device to another without the rights object that is provided by the Rights Issuer. The mobile device is allowed to choose rights from Rights Issuer by opening a browsing session.

Basically, after the presentation of the concepts and models, the main problem is represented by the necessity of developing a secure architecture for key management used within super-distribution business model implemented by Open Platforms DRM Separated Delivery Model.

4 Practical Secure Architecture for the M-Content Distribution and the DRM Key Management

The suggested secure architecture is based on concepts like *electronic signature* and *encryption with symmetric keys* (described in detail by [1], [4], [5]), *SSL* – Secure Sockets Layer and *IP tracking* (described in [4]), *OMA DRM* (described in [3] and in this paper), *XML Signature* and *secure distributed architectures* (described in [6], [7] and [10]).

The secure architecture that puts together all discussed technologies is presented in figure 4.

The central component of the architecture from figure 4 is the DRM Generator and Dispatcher. Two scenarios are presented: the first one pictures a user who buys a media clip and does not have the rights object and the second one is about a user that receives the media clip from a friend via Bluetooth but in order for him to play it he has to receive the rights object. *Scenario I* – the user browser requests via WAP/HTTP (1) from the DRM Generator & Dispatcher Server – DRMGDS a multimedia clip or an application for the mobile device. The DRMGDS sends a download descriptor file back on route (1). The user accepts and the browser sends the second request in order to receive the encrypted media file (DCF). The DRMGDS obtains the content from various Content File Servers from different content providers (2) and interrogates the Keys Management Server – KMS if there is any proper "rights object" information in database (4). *An important note is that the DRMGDS communicates with the KMS via SSL/TSL and with IP filtering restriction – KMS receives requests only from IP – Internet Protocol address of the DRMGDS. Also the requests and responses are encapsulating the XML format of the "rights object" and use XML Signature for authentication.* If the clip user is a new one, then DRMGDS will not find any proper "rights object" at KMS, therefore, DRMGDS generates a symmetric encryption key in order to crypt the content according to AES – Advance Encryption Standard Rijndael 128 bits CBC – Cipher Block Chain. After step (3) from figure 4 the content is encrypted and sent via WAP Gateway to the end-user.

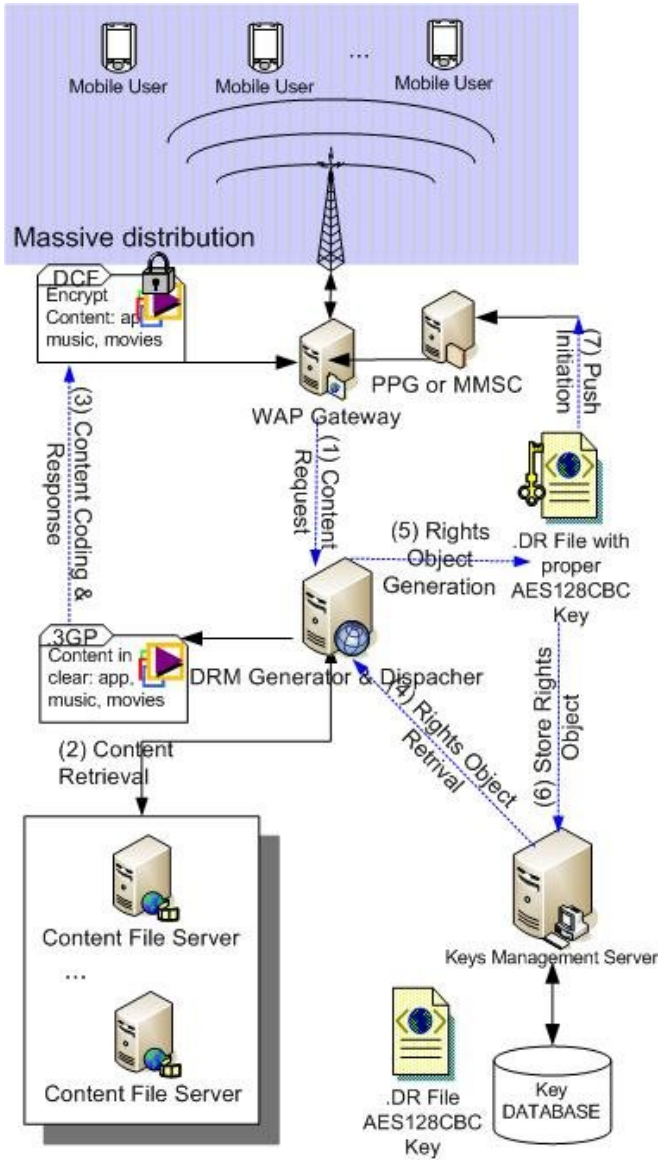

Fig. 4 Proposed Components for Secure Architecture

In the same time, asynchronously, the generated key is embedded into an XML file called "rights object" in step (5) and almost simultaneous is sent via (6) to the KMS to be stored in keys database and is pushed via (7) to Push Proxy Gateway to be received by the end-user.

Scenario II – the user X sends via MMS or Bluetooth to the user Y the media file received in scenario I. The device DRM Agent software of user X filters the media files in keeping with the OMA DRM standard and sends to the user Y the DCF file only – encrypted media file, which contains the URL address to the "rights object". The user Y's browser is instructed by "Rights-Issuer" field from the DCF file and requested via (1) the "rights object". The DRMGDS must find out via (4) the proper "rights object" – DR file – with the proper decryption key or, as an emergency solution if something goes wrong with the KMS, it must send a push message – SL, SI or CO – to redirect automatically the Y's browser in scenario I. If the KMS provides the proper DR file, the DRMGDS just sends a MMS or a push message and ends its job.

In this moment the functionality is established but there might be some security gaps, which could have impact on the QoS – Quality of Service.

A minimal policy package that ensures a reasonable level of security is presented below:

- Physical security – enforce the people that are involved in this system to have access to the components with batches and cards only. Also, the electrical power generator that sustains the electric fluctuations and breakdowns is designed.
- Authentication – all the technical personnel can access components via a Kerberos mechanism with special generated passwords.
- Security standards and protocols – already mentioned, the communication between DRMGDS and KMS is based on IP Selection, SSL/TSL and XML Signature. The media files encryption is based on symmetric key AES-Rijndael algorithm and the format respects OMA DRM 1 standards. The communication between DRMGDS and PPG or MMSC is relay on HTTPS.
- Supervising – all transaction are logged for statistics, billing and clearing procedures with mobile operator and collision avoiding in generating symmetric keys for "rights object" DR files.

5 Conclusions

An alternative solution to the DRM models is to pay for the media object before the user previews it. Another solution is to preview a low-quality level of the media object before downloading process.

The DRM models eliminate some inconveniences of the content delivery and solve two major problems: *a)* the lack of prevention possibilities regarding the transfer of media objects from one device to another; and *b)* the lack of easy and convenient ways in order to preview a media object before it is purchased.

According to the Digital Right Management, the content providers define rules for the media objects usage. The content provider grants the preview rights for the media objects for free and charges the client for a full usage rights only. Therefore, the Digital Rights Management sells rights in order to use the media object and does not sell the media object itself.

The secure architecture for the M-Content Distribution and the DRM Key Management is scalable and can be improved by adding a special module for handling the Microsoft devices. The secure architecture behind the scene is a necessity in order to provide a proper key management for OMA DRM Separated Delivery Model.

References:

[1] Douglas Stinson, *Cryptography – Theory and Practice*–2nd Edition, Chapman & Hall/Crc, NY 2002.
[2] RFC 2045; Multipurpose Internet Mail Extensions – MIME Part 1: Format of Internet Message Bodies, IETF, http://www.ietf.org/rfc/rfc2045.txt
[3] OMA Digital Rights Management 1.0; DRM Content Format, OMA-Download-DRMCF_V1_0-20031113-C, http://www.openmobilealliance.org
[4] William Stallings, *Cryptography and Network Security, 3/E*, Prentice Hall, 2003.
[5] Bruce Schneier, *Applied Cryptography 2nd Edition: protocols, algorithms, and source code in C*, John Wiley & Sons, Inc. Publishing House, New York 1996.
[6] Cristian TOMA, Smart Card Technologies in military information systems, *The 36-th International Scientific Symposium METRA*, May 2005, pp. 500-506.
[7] Cristian TOMA, Secure architecture used in systems of distributed applications, *The 7-th International Conference on Informatics in Economy*, May 2005, pp. 1132-1138.
[8] Ion Ivan, Cristian Toma, Requirements for building distributed informatics applications, *The Automatics and Computer Science Romanian Magazine*, vol. 13, No. 4, November 2003.
[9] Ion Ivan, Paul Pocatilu, Cristian Toma, Alexandru Leau, "e3-com", *Informatică Economică Review*, No. 3(19)/2001, Bucharest 2001.
[10] XML Signature, Full Description Tutorial, http://www.xml.com/pub/a/2001/08/08/xmldsig.html