Secure and Privacy-Preserving DRM for Mobile Devices with Web Service Security*
– An Experience Report –

Carsten Kleiner and Lukas Grittner and Daniel Kadenbach

Abstract Preserving the customer's privacy has to be a major concern when implementing a commercial DRM system. In [12] a privacy-preserving digital rights management (DRM) architecture based on the widely used Open Mobile Alliance (OMA) DRM [17] specification for mobile devices has been suggested. In this paper the design of a possible implementation of the proposed architecture is explained which uses Web Service Security (WSS). This choice has been made since the web services originally designed in the architecture have to meet several security features which are necessary for privacy-preservation. Thus specifically selected WSS features facilitate validation of correctness of the security enhanced concept. This validation is reflected by a detailed security assessment. Moreover a prototypical implementation of privacy-preserving DRM by using a recent WSS implementation (WSS4J) is briefly explained. Finally, along with the experiences from the implementation, a discussion of a potential extension of our suggested architecture and implementation to other DRM systems is given. This discussion also reviews privacy and DRM, both mobile and stationary, in general from a technological point of view. The conclusion is that a similar extension would be possible for all DRM specifications that do not require an online on-access license validation.

1 Introduction

1.1 DRM and Privacy

Among the main issues for criticism of Digital Rights Management (DRM) nowadays are the unnecessary restriction of customers in their rights for fair use of digital

Carsten Kleiner and Lukas Grittner and Daniel Kadenbach
University of Applied Sciences & Arts, Faculty IV, Ricklinger Stadtweg 120, 30459 Hannover, Germany e-mail: ckleiner@acm.org, daniel.kadenbach@fh-hannover.de

* This project has been funded by the Lower Saxony Ministry for Science and Culture under grant no AGIP FA 2005.692.

content as well as privacy concerns regarding information about the user and its usage profile. This criticism is also valid for the currently leading DRM specification which had originally been designed for mobile devices OMA-DRM ([17]).

Since content producers also have a legitimate right to control copyrights on their content, the only solution of this dilemma seems to be designing a privacy-preserving architecture for DRM. In an earlier paper ([12]) we have introduced an architectural concept which is a natural extension to the standard DRM architecture but greatly improves privacy for customers while maintaining the advantages for the content issuers.

To be able to use DRM protected content the user requires licenses, which he can purchase from the license issuer. Present DRM implementations assume a previous registration of the customer at the issuer's side. Thus the customer is forced to abandon private data to the issuer. Any type of misuse of this data when combined with usage profile information by the license issuer is possible. In the DRM architecture that we propose these possible misuse scenarios disappear by adding an intermediary between the customer and license issuer within the purchasing process. This intermediary party is trusted by both the customer and the issuer and conducts the purchase instead of the customer. The focus of our work is on providing a possible technical solution for the privacy problems. We do not consider non-technical influences on the system such as political or legal pressure on the trusted third party to reveal usage information. Preventing such influences is nevertheless a very important issue in order to really preserve the customers privacy. But this is the focus of other work whose results may be complemented with our solution to provide for a fully privacy-preserving solution.

Since the architecture is defined independently of implementation concerns we will propose an implementation via web services with Web Service Security (WSS) enhancements (as specified by OASIS in [16]) that is based on a detailed security assessment of our conceptual architecture in this paper. To proof the feasibility of this approach we have implemented a prototype that will also be described in this article. In addition to this proof-of-concept we will describe our experiences during the implementation phase which will also be helpful for other projects using the same WSS implementation techniques.

After a short description of the OMA DRM architecture which has been used as foundation for our work we will briefly discuss other related work in section 1.3. Section 2 gives an overview of the context and the architecture of our privacy-preserving DRM framework. After that section 3.1 introduces the underlying protocol and the security requirements of the system, followed by a detailed security assessment of the architecture in section 3.2.

Section 4 covers the description of our implementation of the architecture with web service technology using WSS enhancements, while section 5 goes into some of the challenges we encountered. Finally section 6 covers our conclusion of using WSS for the proposed architecture and mentions future work which has to be done as well as future points of investigation.

1.2 OMA DRM

The *OMA* (Open Mobile Alliance) is an association of leading service and product providers in the mobile device sector which specifies interoperable mobile services and tries to establish them as an industry standard. OMA DRM defines a quasi-standard for DRM in the context of mobile devices, but it may in principle be used for DRM on stationary devices as well. Currently the second edition of this specification is the most recent one: OMA DRM v2.0. This enhanced specification is very comprehensive and defines many different use cases.

An important parts of the specification is the definitions of the *ROAP - Rights Object Acquisition Protocol* ([17]). It defines an XML-based communication protocol which is used for the communication between the customer's device and the rights issuer, e.g. to acquire rights objects. Users are purchasing permissions embodied in so-called Rights Objects and the Rights Objects after being transferred to the customer's device need to be handled in a secure and un-compromising manner on the device.

Certificates are used to identify the mobile devices as well as the issuer and hybrid encryption techniques are used to secure the licenses during transmission. For this the public and private key corresponding to the certificate are used to secure a symmetric content key which is then used to decrypt the content on access. The device owner is neither allowed nor required to access the key for the content which is encrypted within the rights object directly. On mobile devices supporting OMA DRM this is controlled by a local DRM-management program (a so-called trusted DRM user agent).

Licenses provided in rights objects are bound to a certain device or set of devices and are thus not exchangeable. This is important for the privacy-preserving extension explained in this paper, since the content issuer may thus transfer the license to the bank without knowing the final client without having to fear that the license is misused (at least the danger is not bigger than it is nowadays without privacy considerations). On current mobile devices trusted DRM agents exist; on stationary hardware such a trusted agent is assumed to exist for OMA DRM to work, but this is currently not the case. Work towards achieving such a software component is underway (e.g. by using trusted computing techniques), but this is not the focus of our work. In line with OMA DRM we base our work on the existence of a trusted DRM agent on client side.

1.3 Related Work

In an earlier paper ([8]) we have already shown how to extend OMA DRM to use web services for communication instead of the proprietary ROAP protocol.

A general introduction to WSS standards can be found in [19] or the well-known original standards documents (e.g. [16]). Standards are a good foundation for building applications, but they have to be employed in a correct and reasonable way.

Our goal was to define all the required details for a DRM web services scenario to prove that the WSS standard is in fact sufficient for these requirements. A security architecture for DRM without web services has been described in [13].

In [6] it is shown how DRM may be used to share personal digital content among a fixed number of well-known users. Since content is just shared in that setting there is no purchase to be conducted which simplifies the process. process. The authors assume that DRM is used to detect illegally published content on the sharing portals.

How SOAP messages can be secured against XML rewriting has been shown in [18], but this security measure is not sufficient for our application. In a policy based setting this has been described in [3]. Due to the work presented in [5] performance problems by adding security to the given DRM scenario should not be expected.

Practical experiences with the technologies in our study are rarely reported so far. In [2] aspect-oriented programming is used in conjunction with the web service security for Java (WSS4J, for more details see section 4.2) implementation and in [7] the technology is used for secure key exchange in web services which is part of our DRM purchase scenario.

A process to incorporate security in general web service design has been described in [10, 9]; these processes have not been used in our work, since we designed the services for the specific DRM scenario only. In a future implementation in an industrial setting the guidelines mentioned in these references should also be considered additionally as well as the ones described in [11]. These guidelines can be understood as complementing our service design.

2 Concept and Architecture

The architecture for a privacy-preserving DRM has been described in detail in [12] already. Hence in this paper only a short description of the architecture is given.

The scenario contains three parties which communicate among each other. These three are an online issuer of DRM secured digital content, a customer and a trusted bank. The purpose of this system is to preserve the privacy and anonymity of the customer towards the content issuer. In current DRM implementations (e.g. based on OMA DRM [17]) customers are forced to register at the issuer to be able to buy the offered digital content. In our implementation the purchasing activity is redirected over a trusted third party. This trusted party is a bank in our case because this way the customer does not have to expose additional private data since he already has an existing account at the bank and also one can assume that the issuers will trust financial institutions like banks.

We introduced a communication protocol which may be used for acquisition of a license in a standard user buy process (i. e. a user purchasing a license for a specific content object). The protocol consists of six messages required for communication among the parties involved. Prerequisite for conducting this protocol is a successful download of a protected (encrypted) content from the issuer's online shop or through some sort of super distribution like peer-to-peer networks. Super distribution refers

to a process where a customer does not obtain the content from the provider directly but rather from some other customer; this distribution mechanism is very important for commercial success. A content object contains an ID which is later used to identify the content and acquire the corresponding licenses for this content. This is not related to privacy issues since the download of the content may occur much earlier than the acquisition of the license; also the content may have been obtained by super distribution without ever contacting the issuer's portal.

This is the starting point of our protocol: a user has obtained a protected content object from some source and now wants to acquire a description of all available licenses for this content object. Therefore he sends a message to the issuer by which he acquires a description of all available licenses for the given content and receives them within the response message. Details on message content have been described in our previous work (see above). After choosing a license the customer transmits a purchase order to his bank, which in return conducts the purchase at the issuer in place of her/him. Because the issuer trusts the bank he provides the bank with the acquired license and receives the corresponding amount of money transferred to his account later on. The bank in turn forwards the license to the customer so he can decrypt and use the content according to the license. During the purchasing action the issuer did not get any information about the customer, but was sure to be paid for her/his product. Figure 1 shows the described scenario.

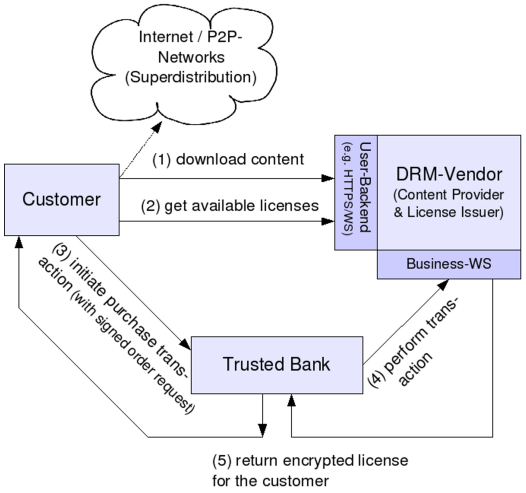

Fig. 1 System architecture for privacy-preserving DRM

The anonymity of the customer towards the issuer may be preserved in many ways in this architecture. Firstly the only information about the customer that the issuer could obtain is the IP address of the device that requests the different licensing options. Thus any well-known approach to achieve anonymity of the IP address may be used here as well, e.g. dynamic IP addresses and an access proxy that hides the

true address of the client from the provider. Since these techniques are not specific to our scenario and also well-known we do not discuss these in more detail here. Secondly the content issuer and license provider do not have to be the same party in OMA DRM. The license issuer only knows the content ID, but possibly does not know more about the content. Finally the time between obtaining the content, obtaining the available licensing options and requesting the usage permission may not be predicted by the issuer. Thus even if the downloads of content or licensing options would be related to IP addresses (which would generate huge amounts of mostly useless data due to usage of shielded IP addresses) the issuer is not able to relate these to the actual license requests since the time lag between the first and second events may not be predicted and during license requests the communication is with the trusted thrid party. Actually in the cases of superdistribution (one customer passes the encrypted content object to another) or license upgrade (a customer obtains a new license because the old one has expired) the customer never downloads the content directly from the issuer so that no correlation between the IP address for download and requestor of a license may be guessed anyway.

The architecture and the defined protocol guarantee the protection of user privacy and the reliability of the license orders. However, the authentication of the customer towards the bank's service or of the bank towards the issuer's service, the confidentiality and the integrity of the messages is not addressed by the protocol itself and thus need to be guaranteed by the implementation which we describe later in this article. Since we will implement the system with web services, we can utilize enhancements like WSS to handle these requirements.

3 Security Issues

3.1 Security Requirements

In the previous section we have mentioned that the protocol per se preserves the privacy of the customer, but the messages themselves are not secured against wire tapping and other attacks during transmission. They could also be spoofed by anyone. Thus they have to be protected against all possible attacks, a more detailed security assessment is in section 3.2.

Figure 2 shows the exchanged messages and the chosen techniques for securing them. The message details are not relevant in this context, we can assume that we just have these 6 messages we want to exchange. We keep in mind that our implementation of the communication protocol will be done using web services. The first communication between the customer and the issuer for the acquisition of the license list should be protected by the use of the HTTPS protocol. As it is well-known this protocol allows the authentication of one or both communication partners, in our case the authentication of the issuer is sufficient and all we want.

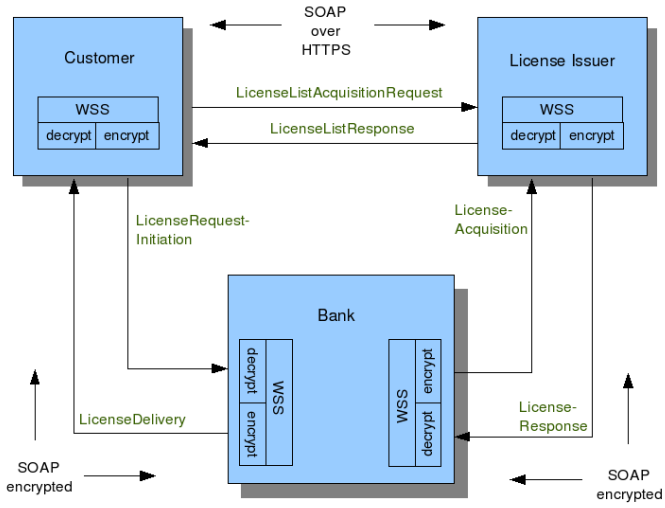

Fig. 2 Message flow in privacy-preserving DRM

Also the encryption of the communication is desired. A requirement is that the issuer owns a certificate.

Not as simple as the requirements for this communication are the conditions for the communication between the customer and the bank. At first the bank should accept only requests from its customers and attacks or malformed messages should be dropped efficiently without bothering the service. Therefore the customer has to authenticate her before requesting a license acquisition. Moreover customers have to encrypt the message in a way the bank and only the bank can decrypt. A solution for this could be the use of certificates on both sides. In conjunction with WSS these certificates can be used to sign and encrypt the exchanged SOAP messages which would solve the security problems. Through the encryption the message is only readable by the receiver and the signature ensures that the message's sender is the one he pretends to be. Additionally a timestamp can be used so that replay attacks are not possible anymore. Using WSS in this way the customer can also be sure talking to her bank and not to an attacker who pretends to be her bank.

Similar to this communication is the message flow between the bank and the issuer. It is important that the sender of a License-Acquisition message is really an official bank. And the receiver of that message is a registered online issuer. Therefore both communicating parties have to authenticate each other to be sure that the transaction is correct. What we need in this case are also certificates that both parties trust. WSS is then used to sign, encrypt and timestamp the SOAP messages similar to the other case.

More detailed information about the requirements of the certificate and its generation is given in section 4.2. The initial setup of the system, i.e. the distribution of the

certificates required, is not any different from the initial setup of any system using a PKI. The certificates should as usual be distributed over a different communication channel than the DRM system they are used for. If additional security measures for security token exchange and trusted web services are required (this should be considered application dependent in our opinion, but is probably desirable) additional web services specifications such as WS-Trust [14] may be integrated.

3.2 Security Assessment

A discussion on user privacy has already been performed in section 2 since it is already observed by the architecture itself. In order to ensure the desired security of the overall system and to be sure that correct and sufficient security technologies have been chosen for the prototype described later, a security assessment has been conducted. Several threats to the aforementioned architecture are imaginable:

- **Information about orders or customer-issuer relationships could be wiretapped** The transferred information in all cases is encrypted with public keys in a way so that only the verified receiver is able to decrypt it. Thus message confidentiality is ensured. A relation between a customer and an issuer can only hardly be constructed by traffic analysis as long as the bank is not compromised. Obviously a compromised bank would make the whole system useless and thus is extremely important here. But assuming a non-compromised bank seems sufficiently reasonable here; see section 6 for a more detailed discussion. To make it even harder for an observer the bank could behave like a Mix [4] and collect a number of purchase orders, mix them and bring them to the same length and then transmit them to the issuers.
- **The customer's authentication data could be stolen** Similar to stealing the PIN of a cash card this is a worst case scenario for the customer since an attacker could buy as much as the customer's bank account allows. Therefore the following countermeasures are in place: The customer has the option to revoke a stolen certificate at the issuing bank and thus make it worthless once he is aware of the stolen authentication data. Since misuse might have already occured at that time, even stronger security is used in our concept additionally: all communication regarding the authentication process in our architecture is encrypted, so that only the verified bank can decrypt the information. It is also protected against replay attacks by inserting timestamps. Thus it is impossible for an attacker to acquire the authentication credentials of the customer by wiretapping the communication between customer and bank, as long as the encryption of the messages cannot be broken. Since the encryption of Web Service Security relies on well-known and tested encryption schemes, this is not a problem. Because of the timestamp encrypted within the message an attacker would be unable to perform replay attacks, thus pretending the customer wanting to execute a purchase transaction several times. Finally, the attacker is unable to pretend to be the bank and thus obtain authentication data of the customer, because certificates

are used to authenticate customer and bank to each other. We can assume that it is impossible for an attacker to acquire the required private key of the bank.
Apart from the communication path the attacker could also try to gain physical access to the customers authentication data: steal customer's certificate and private key directly from the device. To additionally prevent this an additional password should be used, by which the authentication data is protected and which has to be entered by the user any time the certificate or key is used. If desired this feature is supported by our architecture and implementation.

- **The orders could be manipulated** To ensure the integrity of the transferred data, all messages are signed by their senders, so without knowing the corresponding private keys, which should be kept safe, it is impossible to manipulate the content of the messages.
- **License issuer could be spoofed by the bank by using a single license for several customers** Firstly the bank is used as a *trusted* third party, thus it is not expected to spoof anyone in the system. But even if it would do so, since the purchase order is transferred from customer to bank with a specific session key issued by the customer which is later used by the issuer to encrypt the license before it is transferred back to the bank, the bank never has access to the unencrypted license. Thus the bank can't use a license for multiple different customers without knowledge of the issuer or at least the licenses would be worthless since the other customers do not know how to decrypt them.
- **The customer could be spoofed by the license issuer by sending and billing unrequested licenses** The bank acts as *trusted* third party and fully controls the billing process such that only valid requests from customers are processed. The issuer does not have any influence on which license option is used and thus what amount is billed. The license issuer cannot act as a customer by forging purchase orders, since it would need the customer's authentication data (see above).
- **The bank could conduct purchases unknown to the customer** This misuse scenario is already handled properly by our architecture, because every purchase order has to be signed by the corresponding customer, and the certificate generation for the customer certificates should be implemented such that only the customers know their private keys. Since signed purchase orders have to be stored by the bank as evidence, it cannot forge purchase orders of its customers. Also the bank as trusted party should not act abusively anyway.

4 Prototypical Implementation

For the implementation we used the programming language Java, in particular the *JDK 1.6*. As described above the messages have been translated to web services, which allows the usage of web service enhancements like WSS for securing the messages. For the web service part we used *Apache Tomcat 6.0* as application server with *Apache Axis 1.4* as web service module. The WSS for Java (*WSS4J*) implementation in version 1.5.1 was the WSS implementation we used.

4.1 Implementation of Components

Bank The bank is acting as a mediator in the conceptual architecture. Technically speaking the bank is divided into two components. The first component is the customer web service which is contacted by the customer and accepts license requests. The applied interface is shown in listing 3 (all listings are in the appendix).

The web service itself only accepts the requests from the customer and then delivers them to the main service running within the bank architecture, which is the second component. Its duties are verifying the license order, checking the customer's account and conducting the license acquisition on behalf of the customer.

Issuer On the issuer's side there are the two tasks *license acquisition* and *license list acquisition*. Because of their independence they are separated into two services, which can be hosted on different servers for several reasons (e.g. scaling) in practice. The first one is the service for the customer modeled by the interface in listing 4.

For a given content ID a list of available licenses is delivered; the implementation of class `LicenceListAcquisitionResponse` is shown in listing 2. The class has two attributes, one with an array of `License` objects and one which contains the issuers certificate encoded in bytes. This certificate can then be loaded on the customer's side and be checked for validity. If successful it can be used for verifying the license offers which are signed by the issuers certificate.

Furthermore on the issuer's side there exists the service for the bank, which is modeled as shown in listing 5.

As would be expected by the designed architecture the exchanged parameters `LicenseAcquisitionRequest` and `LicenseAcquisitionResponse` contain the required attributes for conducting the acquisition of the licenses. This covers the chosen license description signed by the bank and the issuer, the bank's certificate and a session key which has been generated by the customer. The detailed structure of the two messages is shown in listings 6 and 7.

Since we focused on the WSS part of the DRM architecture, the DRM system itself was not implemented by us. We rather used an already available implementation of an OMA-DRM 2.0 conforming system as a backend for our license issuer. This by itself is neither using web services, nor is secure or privacy-preserving.

Customer The implementation of the customer client application in our prototype works over the console like a stub only. In a real-world scenario a user-friendly client application would be developed which would be installed on the customer's mobile device.

Our stub works as follows: beginning with the prompt for a content ID the program performs a license list acquisition and verifies the issuer's signatures. After that it presents the available licenses to the customer. When the customer chooses to initiate a concrete license acquisition, a session key for AES encryption is generated and a license request is sent to the bank. If the request is successful, the encrypted license is delivered to the device and then decrypted with the session key. Then the license is ready to be used by the DRM client on the customer's mobile device.

4.2 WSS4J

WSS4J is an implementation of WSS [15] in Java. WSS offers several actions which can be performed to the transmitted SOAP message. These are: Sign the message, encrypt the message, authorize via username/token or just add a timestamp. These actions may also be combined. The choice of the features and their configuration is done via deployment descriptors already known from standard web services. These descriptors are used to deploy a web service within an application server and define the available interfaces and how type mapping is done. For web services in conjunction with WSS4J both client and server need such a deployment descriptor.

Chosen preferences: For our purpose we have chosen to use the possibilities *Encryption*, *Signature* and *Timestamp* as actions. These three actions guarantee that the transmitted message is secured against being read by unauthorized persons and being manipulated unknown to the receiver during transmission. Finally the timestamp avoids the success of replay attacks. For more details see section 3.2.

PKI: WSS4J works with certificates to realize the necessary features. In section 5 we will generate two PKIs for the purpose of encryption and authentication. Those PKIs can be used in WSS4J for the mentioned purposes. We will generate two PKIs because in reality we have to distinguish between two different kinds of PKIs. In the first case we want to sign the licence descriptions by the issuer to make the offer reliable. For this reason the certification authority (CA) which issued the content issuer's certificate has to be trusted by the client. So the CA has to be a well known and accredited company for such responsibilities. In a similar way the bank has to prove the customer to be authentic and also it has to present evidence to the issuer that he communicates with a real bank and will get the funds required for the transaction. The second case deals with the communication between the customer and the issuer, especially with the authentication of the customer towards the bank. For this matter it is totally adequate to create a bank internal PKI. So the customer certificates can be issued by the bank itself.

```
<requestFlow>
  ...
</requestFlow>
<responseFlow>
  <handler type="java:org.apache.ws.axis.security.WSDoAllSender">
    <parameter name="action" value="Signature␣Encrypt␣Timestamp" />
    <parameter name="user" value="bank" />
    <parameter name="passwordCallbackClass" value="common.PWBankCallback" />
    <parameter name="signatureKeyIdentifier" value="DirectReference" />
    <parameter name="signaturePropFile" value="bankServer_security.properties"/>
    <parameter name="encryptionKeyIdentifier" value="SKIKeyIdentifier" />
    <parameter name="encryptionSymAlgorithm"
               value="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
    <parameter name="encryptionUser" value="useReqSigCert" />
  </handler>
</responseFlow>
```

Listing 1 Part of deployment descriptor of the bank server

Deployment example: In order to use WSS for the communication between the customer and the issuer we have to enhance the existing Deployment Descriptor of

the server side (bank). More precisely the existing element `<service>` has to be enhanced with the two elements `<requestFlow>` and `<responseFlow>` which (as the names suggest) specify the actions for the service request and service response messages. In listing 1 part of the descriptor for the bank server is shown.

For the request messages, which are received by the bank server from the bank's customers, the actions *Signature, Encrypt* and *Timestamp* are specified as mentioned in section 3.1 already. Within the file *bankServer_security.properties* we specified the keystore, keystore type and other similar properties.

Among the parameters for the outgoing messages the more interesting ones are `signatureKeyIdentifier` and `encryptionUser`. The first parameter specifies how the receiver will have to verify the signature. In our case the value *DirectReference* means that the certificate is sent in conjunction with the message to the receiver. The advantage is that the receiver does not have to have the sender's certificate within its keystore but only the issuers certificate. The value for the parameter `encryptionUser` is set to *useReqSigCert* meaning that for encrypting a message the same certificate as the one signing the received message should be used. The parameter `passwordCallbackClass` specifies the name of the class used to acquire a password for the user's private key specified in the deployment descriptor; this class will retrieve the password out of a sufficiently secure place.

5 Challenges in Implementation

Apart from the typical problems with rather new software libraries such as lack of documentation and tutorials or remaining bugs, we consider the extensive deployment scenario to deserve some attention. As it is known the services have to be deployed to a web service capable web server. Therefore files have to be copied, stubs to be generated, jarfiles to be created and so on. This process can be very fault-prone, so we suggest to use a professional build environment (e.g. Apache Ant) containing several targets to e. g. build the bank's service, build both of the issuer's services or generate the web service description file for the particular services.

Crypto-Provider: There exist several libraries for handling security topics such as symmetric and asymmetric encryption or signatures. Unfortunately there are incompatibilities between these libraries. While running the secured web services some cipher algorithms did not work properly because the crypto-providers used on client and server side were different. After configuring both to use the Bouncy Castle (`http://www.bouncycastle.org`) implementation as the default provider the ciphers worked well.

Certificate Generation: For our prototype we generated all required certificates by ourselves. At first we generated a sample root certification authority, which we then used to issue certificates for the bank and the issuer. Furthermore we generated a second certification authority just for the purpose to issue customer certificates, which are only valid within the communication of a bank and its customers. Figure 3 illustrates the connection between the several certificates.

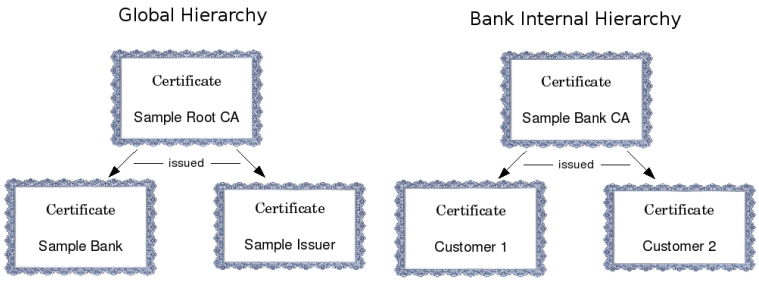

Fig. 3 Certificate hierarchy used for privacy-preserving DRM

For generation of the certificates we mainly used OpenSSL [20], which contains tested C libraries for encryption and some command line tools to create symmetric or asymmetric keys which may be used for encryption or signature; also the tools may be used to generate certificates and sign certification requests in a comfortable way. E.g. a certification request can be executed with the following command:

```
openssl req -x509 -newkey rsa:2048 -keyout cakey.pem -out cacert.pem
```

This command writes the request into the file `cacert.pem` and moreover a private key into the file `cakey.pem`. Furthermore we imported the certificates which were encoded in PEM-format to Java Keystores.

6 Conclusion & Future Work

6.1 Conclusion

By using web services in conjunction with WSS features we were able to fulfil the security requirements for our privacy-preserving DRM architecture. The prototypical implementation underlines that such an architecture may be implemented. In particular the following three steps were necessary: firstly transform OMA DRM to use web services as communication protocol, secondly extend OMA DRM to use a more privacy-preserving architecture and finally use WSS in addition to include necessary security requirements. In the prototype we used only license acquisition.

At the time of building our prototype the WSS4J implementation used does not seem to be fully mature. Since the WSS specification itself is nevertheless fully sufficient, it will only take short time until WSS4J may be used in production.

Our architecture for privacy preservation builds on the straightforward idea of incorporating a third trusted party into the scenario. This is a very natural fit for the DRM situation in our view, since billing has to be taken care of in a DRM context anyway and this is usually not part of the current specifications (e.g. OMA

DRM). Thus it would not only make the specifications more complete, but offer a good chance to take care of privacy concerns in a DRM context in a very fitting and helpful way, since the inclusion of the trusted bank provides advantages for both customers (privacy) as well as license issuers (billing and reliability).

Of course our architecture only works if the bank can really be trusted. If there is any chance that the bank might spoof any of the other parties the security of the system is lost to a high degree. Also the privacy of users depends on the trustfulness of the bank. If there is a chance that e.g. by legal pressure anyone might obtain information that is usually only known by the bank, then the privacy of users would be in danger. We feel that this is not a big restriction, since the trusted third party in our scenario is probably a *real* bank. If pressure on a real bank may lead to privacy concerns, there are more serious privacy issues than DRM usage profiles.

6.2 Future Work

Still work has to be done to drive our implementation from a prototypical proof of concept to a real-world usable application scenario, so that it will become possible to investigate the system and its usability, security, stability, scalability and other aspects in further detail. Also, other DRM processes than license acquisition have to be investigated in the same way to come up with a complete implementation. Conceptually we expect to have solved all problems in our prototype already.

As always the case with prototype implementations they only reflect a snapshot. Thus another interesting point would be to investigate more recent web service and WSS implementations, which were not available at the time of our implementation, especially concerning their improved usability. For example the WSS4J 1.5.2 version or the promising Axis 2 [1] implementation.

Finally a possible extension and transfer of our privacy and security measures to other DRM systems than OMA DRM would have to be examined. We expect that as long as the communication in that system is carried out by web services (or a different means which offers similar security features) the same approach should also work for other DRM systems. This claim is valid for DRM which uses a trusted agent on the customer side for license checking (such as OMA DRM or HDCP).

If a DRM specification requires a per use activation or online license checking on access of the protected material, the architecture does not work directly anymore. In these cases it is much more difficult to guarantee user privacy, since the online checking also has to be executed in a privacy preserving manner. Thus this step would also have to be performed by a party other than the issuer (maybe another trusted party) making the whole scenario even more complex. Also the usage information of customers would be much more valuable and thus in danger since detailed usage profiles are possible. In OMA DRM only license acquisition but not actual use might be monitored without privacy preserving extensions. Consequently nowadays only DRM specifications based on trusted agents for license checking seem to prevail which is good news from a privacy perspective.

References

1. Apache Software Foundation: Axis 2, Version 1.2 (2007). http://ws.apache.org/axis2/
2. Baligand, F., Monfort, V.: A concrete solution for web services adaptability using policies and aspects. In: ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing, pp. 134–142. ACM Press, New York, NY, USA (2004)
3. Bhargavan, K., Fournet, C., Gordon, A.D., O'Shea, G.: An advisor for web services security policies. In: SWS05: Proc. of the 2nd WS on Secure web services, pp. 1–9. ACM Press (2005)
4. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms (1981). http://world.std.com/~franl/crypto/chaum-acm-1981.html
5. Chen, S., Zic, J., Tang, K., Levy, D.: Performance evaluation and modeling of web services security. In: Proceedings of the International Conference on Web Services, ICWS 2007, pp. 431 – 438. IEEE Computer Society, Salt Lake City, UT, USA (2007)
6. Conrado, C., Petkovic, M., van der Veen, M., van der Velde, W.: Controlled sharing of personal content using digital rights management. In: E. Fernández-Medina, J.C. Hernández, L.J. García (eds.) WOSIS, pp. 173–185. INSTICC Press (2005)
7. Fang, L., Meder, S., Chevassut, O., Siebenlist, F.: Secure password-based authenticated key exchange for web services. In: SWS '04: Proceedings of the 2004 workshop on Secure web service, pp. 9–15. ACM Press, New York, NY, USA (2004)
8. Grittner, L., Kleiner, C., Kadenbach, D.: Implementing oma drm using web services. In: Proc. of the 2nd Int. WS on Mobile Services-oriented Architectures and Ontologies (MoSO). Mannheim (2007)
9. Gutiérrez, C., Fernández-Medina, E., Piattini, M.: Web services enterprise security architecture: a case study. In: SWS '05: Proceedings of the 2005 workshop on Secure web services, pp. 10–19. ACM Press, New York, NY, USA (2005)
10. Gutierrez, C., Fernandez-Medina, E., Piattini, M.: Pwssec: Process for web services security. In: Proceedings of the International Conference on Web Services, ICWS 2006, pp. 213 – 222. IEEE Computer Society, Chicago, IL, USA (2006)
11. Hepner, M., Gamble, M., Gamble, R.: Forming a security certification enclave for service-oriented architectures. In: Proceedings of the International Conference on Web Services, ICWS 2006, pp. 148 – 155. IEEE Computer Society, Chicago, IL, USA (2006)
12. Kadenbach, D., Kleiner, C., Grittner, L.: A drm architecture for securing user privacy by design. In: Proceedings of the 5th International Workshop on Security in Information Systems, WOSIS 2007, pp. 188 – 195. IEEE Computer Society Press, Madeira, Portugal (2007)
13. Michiels, S., Verslype, K., Joosen, W., Decker, B.D.: Towards a software architecture for drm. In: DRM05: Proc. of the 5th ACM WS on Digital rights management, pp. 65–74. ACM Press (2005)
14. Nadalin, A., Gudgin, M., et. al.: Web services trust language (ws-trust) (2005). http://www.ibm.com/developerworks/library/specification/ws-trust/
15. Nadalin, A., Kaler, C., Monzillo, R., Hallam-Baker, P.: Oasis web service security (2006). http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
16. OASIS: Web services security specification (2006). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
17. Open Mobile Alliance Ltd.: OMA DRM v2.0 (2006). http://www.openmobilealliance.org/release_program/drm_v2_0.html
18. Rahaman, M.A., Schaad, A., Rits, M.: Towards secure soap message exchange in a soa. In: SWS06: Proc. of the 3rd ACM WS on Secure web services, pp. 77–84. ACM Press (2006)
19. Soumeeh, R.E.: Web service security (2006). http://www.st.informatik.tu-darmstadt.de/database/seminars/data/WS-Security.pdf?id=155
20. Young, E.A., Hudson, T.J.: OpenSSL Project (2004). http://www.openssl.org

Appendix: Sample Listings of Prototypical Implementation

In this appendix some sample listings of the prototypical implementation are shown for illustrative purposes. The listings are not meant to give a full reference to the implementation; such a reference may be obtained by contacting the authors directly.

```
public class LicenceListAcquisitionResponse {
  private Licence licenceList[];
  private byte issuerCert[];
  ...
}
```

Listing 2 Structure of `LicenceListAcquisitionResponse`

```
public interface RequestLicenceInterface {
  public LicenceDelivery requestLicence(
    byte encSessionPublicKey[],
    Licence licence, String timestamp,
    byte licenceSignature[],
    String issuerURL );
}
```

Listing 3 `RequestLicense` Service Interface

```
public interface LicenceListAcquisitionInterface {
  public LicenceListAcquisitionResponse
      licenceListAcquisitionRequest( String contentID );
}
```

Listing 4 `LicenseListAcquisition` Service Interface

```
public interface LicenceAcquisitionInterface {
  public LicenceAcquisitionResponse
    licenceAcquisitionRequest(LicenceAcquisitionRequest request);
}
```

Listing 5 LicenseAcquisition Service Interface

```
public class LicenceAcquisitionRequest {
  private Licence selection;
  private byte[] signature; // signature with the banks certificate
  private byte[] publicKey; // certificate of the bank
  private byte[] encSessionPubKey;
  ...
}
```

Listing 6 Structure of `LicenceAcquisitionRequest`

```
public class LicenceAcquisitionResponse {
  private String status;
  private byte[] encryptedLicence;
  private String accountID; // account ID of the issuer
  ...
}
```

Listing 7 Structure of `LicenceAcquisitionResponse`