XML Signatures in an Enterprise Service Bus Environment

Eckehard Hermann

Research & Development XML Integration
Uhlandstraße 12
64297 Darmstadt, Germany
Eckehard.Hermann@softwareag.com

Dieter Kessler

Research & Development XML Integration
Uhlandstraße 12
64297 Darmstadt, Germany
Dieter.Kessler@softwareag.com

Abstract.

The goal of service oriented architectures (SOA) is to allow a message based and loosely coupled interaction between different web services. This approach allows the orchestration of web services in distributed, heterogeneous applications where the different services can be implemented in different programming languages, run on different machines and be based on different protocols. The adoption of web services to integrate systems within an organization and with partners is strongly dependent on the security standards that accompany service oriented architectures (SOA). The XML (Extensible Markup Language) Signature standard plays a key role here. For protecting such a distributed application, XML Signatures are used on several levels and for different challenges, for example to guarantee the integrity and authenticity of the exchanged messages and their authentication information, as well as the audit trails and to provide non-repudiation. The paper describes the role of XML Signatures for protecting Enterprise Service Bus (ESB) based SOA applications.

1. Introduction

1.1 Background - Enterprise Service Bus

The phrase “Enterprise Service Bus” describes an architectural pattern in which applications are created by composing software components together in such a way that complete business processes are reflected. In an ESB, the software components are packaged as high-level “services”, which are platform-neutral and which can be described in terms of the documents which they take as input and the documents

which they produce as output. These input and output documents are formatted as XML. An ESB-based application therefore involves XML documents, being routed from one software component to the next, where each one performs a particular processing step on the document. ESBs differ architecturally from earlier "hub and spoke" integration patterns. The central role within the "hub and spoke" architecture is played by a broker, which connects all services. In contrast, an ESB is a message oriented middleware (MOM) that allows all services to communicate directly with each other. Another difference is that an ESB is independent of the transfer protocol: the SOAP binding defines how the messages are bound to the different protocols such as TCP/IP or HTTP.

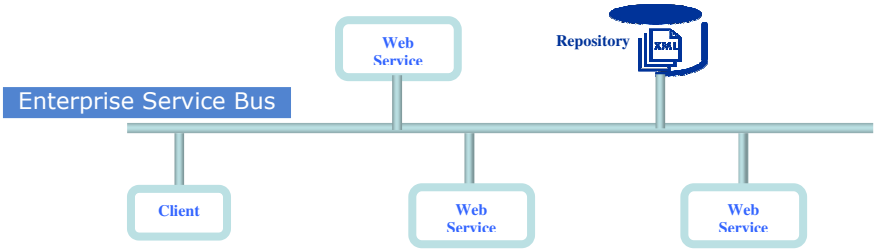

Fig. 1. Orchestration of internal and external services with an ESB

It can be argued that ESB architectures have been in existence for many years, but the addition of XML technologies has made ESBs much easier to implement. These XML technologies include:

- SOAP (Simple Object Access Protocol, since SOAP 2.0 it is no longer an abbreviation): For enveloping XML documents which are exchanged among services in an ESB, and for routing
- WSDL (Web Services Description Language): For describing the different services, orchestrated by the ESB
- XML Schema: For describing XML documents used by an ESB
- XSLT (Extensible Stylesheet Language Transformation): For converting an XML document from one format to another
- XML Signature: For securing the service oriented architecture on different levels

1.2 XML Signature

The data to be signed with an XML Signature is referenced via a URI with the help of Reference elements which are part of a SignedInfo element of an XML signature. The data to be signed can be transformed, using the Transforms element, prior to forming the hash value for the canonical form of the referenced data. This hash value which is subsequently computed for the referenced and transformed data is entered in the SignedInfo part of the XML signatures as the DigestValue element. When the DigestValue has been generated for all references, the canonical form is created from

the SignedInfo element of the Signature element and a second hash value is computed for the result. This hash value is encrypted using the signer's private key. It constitutes the actual digital signature and is attached to the XML signatures as the SignatureValue. All referenced documents are thereby endowed with one digital signature [1].

```
<Signature ID?>
   <SignedInfo>
     <CanonicalizationMethod/>
     <SignatureMethod/>
     (<Reference URI? >
       (<Transforms>)?
       <DigestMethod>
       <DigestValue>
     </Reference>)+
   </SignedInfo>
   <SignatureValue>
  (<KeyInfo>)?
  (<Object ID?>)*
</Signature>
```

Fig. 2. XML Digital Signature Syntax [2]

1.3 Secure the Service Oriented Architecture

Like other applications, a service oriented architecture is protected by addressing the following requirements and finding an adequate answer to the related questions:

- **Authentication** - Who is attempting to gain access?
- **Authorisation** - Which resources is this client allowed to access?
- **Integrity** - Has the data or system been tampered with?
- **Confidentiality** - Can the data be read while in transit or storage?
- **Non-repudiation** - Can a sender deny having sent a message?
- **Auditing** - Is there a record of client access to data?
- **Availability** - Is this system vulnerable to a denial-of-service attack?

For securing a SOA, XML Signature is a key concept to address the authentication, integrity and non-repudiation topics directly, and authorization and auditing indirectly. In the following the role of XML Signature in the different areas is discussed.

2. Importance of XML Signature

2.1 Motivation

As mentioned above, Enterprise Service Bus based applications orchestrate different web services into one distributed application. Each request is processed as one business transaction. To be able to analyse and reproduce the different steps in case of an attack, (during a business transaction) a legally relevant audit trail is important. It can also be used as input information for intrusion detection systems to prevent future attacks after the same schema. To give an audit trail legal relevance, it is necessary to know when and by whom the trails were created and if it was possible to manipulate the audit trail. A whole new dimension is added by the cascading nature of applications, which is given in case of an ESB based SOA. Each web service participating in a business transaction, as well as the ESB itself, writes its own audit trail, each of which must be protected.

Normally web services are not directly used by a human user. At the very least, there is a client application between the web service and the user, in more complex scenarios like an ESB based application, there are a web service client, the ESB and probably several cascading web services. In the worst case, each of these services needs to authenticate and authorize the user. To allow the services to check the authentication result and to authorize the requesting user, but on the other hand free the user from having to logon multiple times (which would require a distributed administration of authorization rules), a single sign on mechanism and centralized point of administering access rights is necessary.

In a distributed environment where messages travel from the sender to the receiver via several hosts and services, it is important for the receiver to be able to prove that the message arrived unchanged, by whom it was sent, and perhaps at what exact time it was sent or arrived.

Digital signatures and especially XML Signature can be a solution for these issues, as discussed in the following.

2.2 Identity Federation in a Distributed Web Services Environment

One of the most critical topics of a distributed heterogeneous application, especially if legacy systems are involved, is the authentication and authorization of the requesting client. Independent applications and components that are wrapped by a web service and which are orchestrated to one distributed web service application normally have their own authentication and authorization behavior, running in their own independent domain.

2.2.1 The Identity Provider

The orchestration of these components via web services also includes the orchestration of the authentication and probably the centralisation of the access right administration, to allow the definition of overall security policies. The problem of different domains where user Bob in domain A is not equal to user Bob in Domain B can be solved by the introduction of a trusted third party, the Identity Provider, which plays the role of a central point of authentication and user management. All the different participating web services have to trust the Identity Provider and its user and role mapping decisions. The Identity Provider has to know that the identity of a client e.g. “Joe” authenticated as “Bob” in domain A is the “Steve” in domain B.

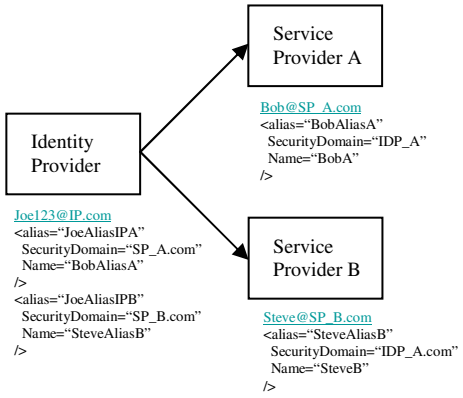

Fig. 3. Identity provider and multiple service providers as identity federation [4]

The authentication has to be done by the Identity Provider. After a successful authentication, the Identity Provider provides the identity of the authenticated user in the different domain. This identity information can now be placed as part of the message, sent to the distributed web service application, which allows the participating web services to find out the user’s identity and authorize the user depending on his local access rights. To exchange messages between web services, the SOAP protocol, defined in XML is used. A SOAP message consists of a header and content section. The SOAP header contains meta information, necessary for routing and processing the data provided in the content section of the SOAP message. The distinction between the content and meta data sections of a SOAP message makes it possible to place the identity information as part of the SOAP Header, which is defined by the Web Service Security standard [7].

2.2.2 Identity Federation

The complete concept of this single sign-on approach is based on the trust relationship between the different web services and the Identity Provider and the integrity and authenticity of the identity information, placed in the messages. To guarantee integrity and authenticity, the identity data is digitally signed by the Identity Provider. Because the protocol that is used to exchange the message and the message

themselves are in XML, the signature created by the Identity Provider is also an XML Signature. To validate the integrity and authenticity of the identity information, the web services have to validate the XML Signature signing the identity information and created by the Identity Provider. Beside the identity information, the XML Signature is also placed in the message as part of the SOAP header. [7]

2.3 Message Integrity and Non-Repudiation

In an open architecture like the SOA, it is vital to protect the messages that are being transmitted from client to server and further from the server to other services within or even outside a closed network. Because of the human-readable format of XML messages, attackers are very much attracted. When orders are being passed over the internet, the issuer, the order object and sometimes the bank details are transmitted. If this happened over an unprotected line, attackers would have a very easy job in forging and spoofing messages either from the client or from the service.

It is therefore very important to have a tool to guarantee that the message was not modified in transit and that each party can be sure that no other fraudulent party is participating. One of the first countermeasures is the usage of SSL [10].

- SSL allows the client to determine that the service is the real service that he is supposed to communicate with. The SSL server (i.e. the service provider) sends his certificate, which is checked by the client (application).
- SSL guarantees message integrity. All traffic between the two TCP endpoints is protected from forgery.
- SSL encrypts all traffic, so that the (even plain text XML) messages cannot be read by unauthorized parties.

SSL is widely available and understood by all major browsers. Also, SSL can easily be configured for web services, and commercial applications usually support it as well. But SSL has one disadvantage, which is that encryption and other security functions are performed only by the two TCP endpoints. If the message needs to be secured further on, then an end-to-end protection like the usage of XML Signature and XML Encryption in the application level is required.

Using XML Signature and XML Encryption on the application level takes up more coding but gives a more finely grained method to guarantee the message integrity, privacy and non-repudiation. The client signs the message content and places the signature into the message header. The references of the XML Signature point to the signed data in the message content. Using the Transforms element, it is possible to protect only parts of the message context or the complete message and in case of a multipart message the attachment. The validity of the signature guarantees the receiving service the integrity and authenticity of the message.

2.4 Signed Audit Trails

One of the most critical issues in terms of the legal requirements is a method that enables the provider of a service to prove that certain activities have taken place and data have been processed in a particular way by the providing service. The best instrument for this task is the signed audit trail.

Audit trails are basically the protocol about what happened at a specific service. Now, when we look at the SOA architecture, we will quickly see that one action triggered by a client can lead to multiple (sub-) actions carried out by other web services. In order to backtrack the end user's transaction accurately, this requires the collaboration of all participating services to add meaningful and important information to this trail. Audit trails are providing the following capabilities:

- They guarantee non-repudiation. I.e., if dealing with a customer order, the manufacturer can prove that a certain order was placed by a certain principal.
- They also give the client a tool by which he can review (double-check) his orders.
- Audit trails should provide enough qualitative information to re-build the business transaction, in case of problems like attacks, disk crashes, breakdowns and outages of the service did happen.

Certain features are required for different levels of trust that an audit trail can provide:

- Identification and authentication of the client. Can the client be identified uniquely and could he authenticate himself in a trustworthy way? This must be logged in case of a non-repudiation requirement.
- The usage of single sign-on. The identity of the caller should be the same for all subsequent web service requests within this transaction. Optionally, each service may log the time and principal identification in case of any doubt.
- If the authenticated client is been given a certain role, this needs to be logged as well. All participating services must state clearly for which account the action was carried out.
- One of the most critical issues is the quality of the signing certificate. Normally, a batch signature with an advanced signature is sufficient. Sometimes there may be the need for a qualified signature.
- In order to trust all related parties, and that all the applications work reliably to a certain degree, is a certification required? This will certainly improve the trustworthiness of the whole application. Still, this is a very costly process and needs to be evaluated thoughtfully.

Because of the complexity of an enterprise service bus, the creation of audit trails is similarly complex. A central utility is a good start to implement signed audit trails.

Within the suggested environment, a native XML server with the cryptographic capability to handle electronic signatures is the ideal solution. In addition to the bus type communication of the application data, each service itself has a direct line to the logging process (i.e. an XML server). Given that the XML server supports

transactions, each (sub-) service carries forward the data required to complete an audit log for the whole service request.

The above suggestion will also work in a widely distributed environment. This is due to the fact that the XML server itself can be configured to act like a protected service.

3. The Software AG ESB approach

The Software AGs Enterprise Service Bus, the Enterprise Service Integrator (ESI), is also confronted with the security issues discussed above.

3.1 Identity Federation

For Software AG, single sign on is a very important topic. One of the most common use cases of the ESI is the web service integration of legacy applications running in a mainframe environment. By default, mainframe applications run in a closed environment with a closed and limited set of users. If such applications are integrated into a distributed web application, accessible via the Inter- or Intranet, a strong security policy has to be defined. Authentication, authorization and user handling are very sensitive topics for such applications. For most of the use cases it is important to authorize the primary user, who initiated the business transaction. Because of the multi-tier architecture of distributed web services applications, authenticating the requesting user is done by the web service client and not in the backend system. In such a case, the mainframe application has to trust the authentication and user's identity information provided by the authenticating entity, as well as the integrity of the transmitted data. A strong trust relationship is given by digitally signing the authentication assertions and the identity data.

The ESI handles signed authentication and authorization assertions and supports a role based access control approach. This requirement is provided by the support of SAML (Secure Assertion Markup Language) [9] based Web Service Security. The security assertions are digitally signed by the Identity Provider and the signature is validated by the ESI and the mainframe application before the requesting user is authorized to execute the request.

3.2 Message Integrity and Non-Repudiation

The ESI demonstrates the design of a middleware that allows companies to implement message integrity with or without the knowledge of the underlying web services. The usage of XML Signatures (i.e. the validation process of incoming messages as well as the automatic signature generation of outgoing responses) can be carried out without the service noticing this additional feature. In case the signature

and data integrity keeps its important role beyond the mere transport and time of conversation, the web service can make active use of the signature features of the ESI.

Furthermore it should be pointed out that in most cases of message authentication a digital signature does not provide enough information. Typically a timestamp needs to be added to the signature. The digitally signed document (or a hash thereof) together with date and time information guarantees for both sender and addressee that a certain message or document was only sent once (not replayed) and, if used with a signed response, to prove to the sender that his original document was not modified in transit plus that it was accepted at a certain time. This "receipt" provides the legal binding to business transactions being originated by the web service request.

3.3 Signed Audit Trails

Most of the various web services within a distributed web service application that participate in processing a request as well as the ESI itself write their own audit trails. To guarantee integrity of the audit trails, the ESI follows two different approaches.

1. At the end of a business transaction all audit trails, written during this business transaction, are referenced and digitally signed with one detached XML Signature.

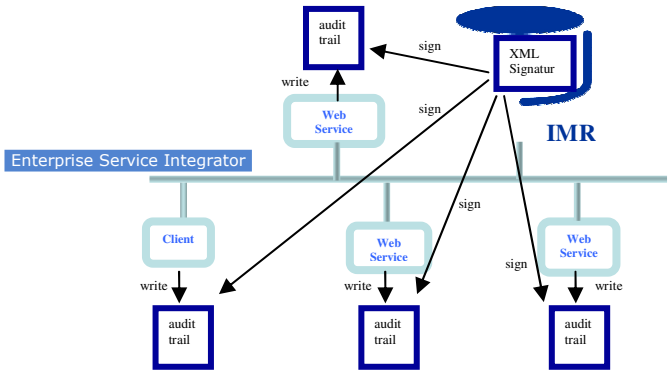

Fig. 4. ESI based application with several audit trail documents

2. A higher degree of security is obtained by writing the audit trail directly into the Integrated Metadata Repository (IMR), which is part of the Enterprise Service Integrator. The locking and security concept of the IMR guarantees protection against parallel manipulation of the audit trails until the transaction is committed. Before the transaction is committed, the audit trail is digitally signed with an XML Signature, containing a timestamp as well.

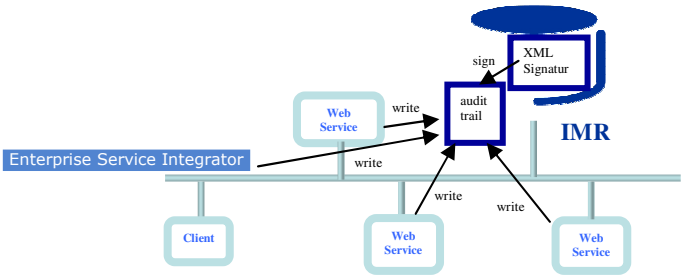

Fig. 5. ESI based application with one central audit trail document

4. Conclusion

In protecting distributed web services applications, XML signatures play a very important role. In case of business transactions single sign on, message integrity and non-repudiation are important security requirements that can be addressed. Other problems can only be solved by using XML Signatures, such as signing the audit trails of several components located at different places and involved in one business transaction with one signature. An additional dimension is introduced in case the signed audit trails have to be archived over a long time. In such cases, XML Signature can also be used to guarantee integrity and non-repudiation for a long time, including the possibility of renewal of the XML Signatures by resigning periodically. An approach is given by the Long-term Archive and Notary Services (LTANS) Internet-Draft [8]. Long-term archiving is one of the next important steps to take, and in which XML Signatures will play a key role.

5. References

[1] René Kollmorgen, Dieter Kessler, Eckehard Hermann, Frank Jung: Digital signatures in XML, http://asia.cnet.com/builder/architect/system/0,39009336,39100045,00.htm
[2] XML Signature Syntax and Processing, http://www.w3.org/TR/xmldsig-core/
[3] Eugene Kuznetsov: XML Web services security best practices, http://www.builderau.com.au/manage/work/0,39024674,39130825,00.htm
[4] Liberty ID_FF Architecture Overview, Version: 1.2-errata-v1.0, https://www.projectliberty.org/specs/draft-liberty-idff-arch-overview-1.2-errata-v1.0.pdf
[5] Software AG ESI Security and SOA Security white papers, http://www.softwareag.com
[6] SOAP Version 1.2 Part 0, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/
[7] OASIS Web Service Security: SOAP Message Security, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
[8] Long-term Archive And Notary Services (LTANS) Internet-Draft, http://ietfreport.isoc.org/ids/draft-ietf-ltans-ers-02.txt
[9] SAML v2.0, OASIS, http://www.oasis-open.org/specs/index.php#samlv2.0
[10] IETF Working Group on Transport Layer Security, http://www1.treese.org/ietf-tls/