WEB SERVICES SECURITY: A PROPOSED ARCHITECTURE FOR INTERDOMAIN TRUST RELATIONSHIP

A Thesis Submitted to
The Graduate School of Engineering and Sciences of
Izmir Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Selim L. Y. HENDRICKSON

November 2006
İZMİR

We approve the thesis of **Selim L Y HENDRICKSON**

Date of Signature

…… **27 November 2006**
Asst. Prof. Dr. Tuğkan TUĞLULAR
Supervisor
Department of Computer Engineering
İzmir İnstitute of Technology

…… **27 November 2006**
Assoc. Prof. Dr. Ahmet KOLTUKSUZ
Department of Computer Engineering
İzmir İnstitute of Technology

…… **27 November 2006**
Prof. Dr. Şaban EREN
Department of Computer Engineering
Ege University

…… **27 November 2006**
Prof. Dr. Kayhan ERCİYEŞ
Head of Department
İzmir İnstitute of Technology

……..
Assoc. Prof. Dr. Semahat ÖZDEMİR
Head of Graduate School

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Assistant Professor Tuğkan TUĞLULAR for not only teaching me everything I know, but also for assisting me with every aspect of my life. I also would like to thank Associate Professor Ahmet KOLTUKSUZ for teaching me to search for different angles in every subject. I would like to offer many thanks to my dear friend and biggest support Fatih ALGAN for everything he has done for me. I'm grateful to all the talented, hard working engineers I had the privilege of working with in the past four years. Last but not the least, I would like to thank my mother not because she has been putting up with me all my life but because I love her madly.

ABSTRACT

WEB SERVICES SECURITY: A PROPOSED ARCHITECTURE FOR INTERDOMAIN TRUST RELATIONSHIP

Web services technology is vulnerable to security threats similar to other technologies which are based on communication over internet. Some applications working over internet typically require strong authentication. The security requirements of a scenario may involve interdomain authentication mechanisms. These domains may be operating using different technologies.

In order to enable such scenarios, we leverage existing approaches with emerging standards and propose an architecture. Our proposed architecture takes advantage of XML technology and emerging SAML standard. The most important aim of the proposed architecture is platform indepedence.

Our proposed architecture includes a Security Token Service and a protocol for communication between token requesters, consumers and issuers. Although, the exact flow of execution depends on the scenario, we believe our approaches can be used as common ground for implementation.

ÖZET

WEB SERVİSLERİ GÜVENLİĞİ: ÇALIŞMA ALANLARI ARASI GÜVEN İLİŞKİSİ İÇİN BİR MİMARİ ÖNERİSİ

Web Servisleri teknolojisi internet üzerinden iletişime dayanan diğer teknolojiler ile aynı güvenlik tehditlerine açıktır. İnternet üzerinden çalışan kimi uygulamalar güçlü kimlik doğrulamasını talep ederler. Bir senaryonun güvenlik talepleri, çalışma alanları arası kimlik doğrulama mekanizmalarına ihtiyaç duyabilir. Bu çalışma alanları farklı teknolojiler kullanarak çalışıyor olabilir.

Bu tip senaryolara imkan sağlamak amacıyla, varolan yaklaşımları yeni gelişmekte olan standartlarla birleştirip bir mimari öneriyoruz. Önerdiğimiz mimari XML teknolojilerinden ve gelişmekte olan SAML standardından faydalanmaktadır. Önerilen mimarinin en temel amacı platform bağımsızlığıdır.

Önerilen mimari bir güvenlik jetonu servisi ve jetonu talep edenler, kullananlar ve yayınlayanlar arasındaki iletişimi sağlamak için bir protokolü içerir. Senaryoların kesin akışı duruma bağlı olarak değişebilir, ancak inancımız yaklaşımlarımızın uygulama geliştirmek için bir temel olarak kullanılabileceği yönündedir.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Web services technology, which enables disparate systems to interoperate at a high level with ease, lacked a common framework for security. Although some attempts have been made to solve this problem, most of these attempts bring a solution using variations of technologies already in use. These techniques, which are actually proven in the field, do not fit with the interoperable and loosely coupled nature of Web services technology. The new emerging Web Service Security (WS-Security) standard, along with other extensions of the WS-* group of technologies aims this lack of a common security framework.

The Web Services Security standard is formed by independent organizations backed by big vendors of the industry such as IBM, Microsoft, RSA, and Verisign. Although, some alignment in vision has been achieved, in practice there is still a hot debate. This debate, although a good thing for advancement of technology, results in a slow penetration of technology. Proven products leveraging these technologies are still missing.

This thesis aims to provide an understanding of this new technology and proposes architecture for a custom scenario, which we believe is comprehensive and extensible. The design and architecture of the case study examined in this thesis, is implemented using a mix of several Web services security specifications. Although there are alternative specifications still competing to become de facto standards, we believe our selection of specifications had recently gained momentum and success in becoming a de facto standard in the near future. The architecture proposed in this thesis, can be extended to achieve to enable larger scenarios or can be downsized and used partially for more common smaller use cases.

CHAPTER 2

WEB SERVICES

2.1. Introduction to Web Services

This chapter introduces Web Services, a technology that enables an application to invoke another application over the Internet leveraging the existing standards and protocols. The formal definition for Web Services as defined by W3C is:

A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards "(WEB_1 2006)".

During the development of this technology, focus was put on making functional building blocks accessible over standard Internet protocols that are independent from platforms and programming languages. Therefore, different software systems, running on different platforms, developed using different technologies can interact if they leverage Web Services technology. These systems can be new applications or existing legacy systems wrapped around with an additional layer to make them Internet-enabled. As a consequence of this, applications which has become services, can rely on and cooperate with each other to achieve new goals.

2.2. A Short History of Web Services

With the spread of Internet, it became clear that the infrastructure that was introduced by Internet could be used not just to retrieve information that was to be presented using a browser (called human-to-application, H2A applications). Rather, there was also an increased demand for application-to-application (A2A) communication using the existing technologies. Existing protocols were used in order to achieve this goal. However, not long after, it became obvious that existing protocols fell

short of accomplishing this goal. The HTTP protocol does not provide for complex applications that arise from A2A scenarios.

In late 1999, an XML based protocol, namely the SOAP protocol was published by the Microsoft Corporation, which could be used for A2A scenarios. IBM Corporation started supporting the SOAP protocol in early 2000 followed by the public acceptance of SOAP by the industry. In May, 2000 the SOAP protocol was submitted to the W3C Consortium and in June, 2003 it was released as a W3C Recommendation.

Numerous other protocols for supporting and extending Web Services development have been published. Some of the most important protocols are, WDSL, UDDI, and the WS-* suite of protocols. We will be examining the WSDL and UDDI protocols briefly in the next section, and WS-Security protocol, which is part of the WS-* suite of protocols, in the fourth chapter.

2.2. Web Services Protocol Stack

Web Services are a set of protocols based on XML. The following protocols formed the initial specification for the Web Services. The lifecycle for a general web service life cycle scenario is presented in Figure 2.1.

Simple Object Access Protocol (SOAP) – The formal set of conventions governing the format and processing rules of a SOAP message. These conventions include the interactions among SOAP nodes generating and accepting SOAP messages for the purpose of exchanging information along a SOAP path "(WEB_1 2006)".

Web Services Description Language (WSDL) - WSDL or Web Services Description language is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate "(WEB_1 2006)".

Universal Discovery, Description, Integration (UDDI) – UDDI is the technical foundation for publication and discovery of Web Services implementations both within and between enterprises "(WEB_2 2006)".

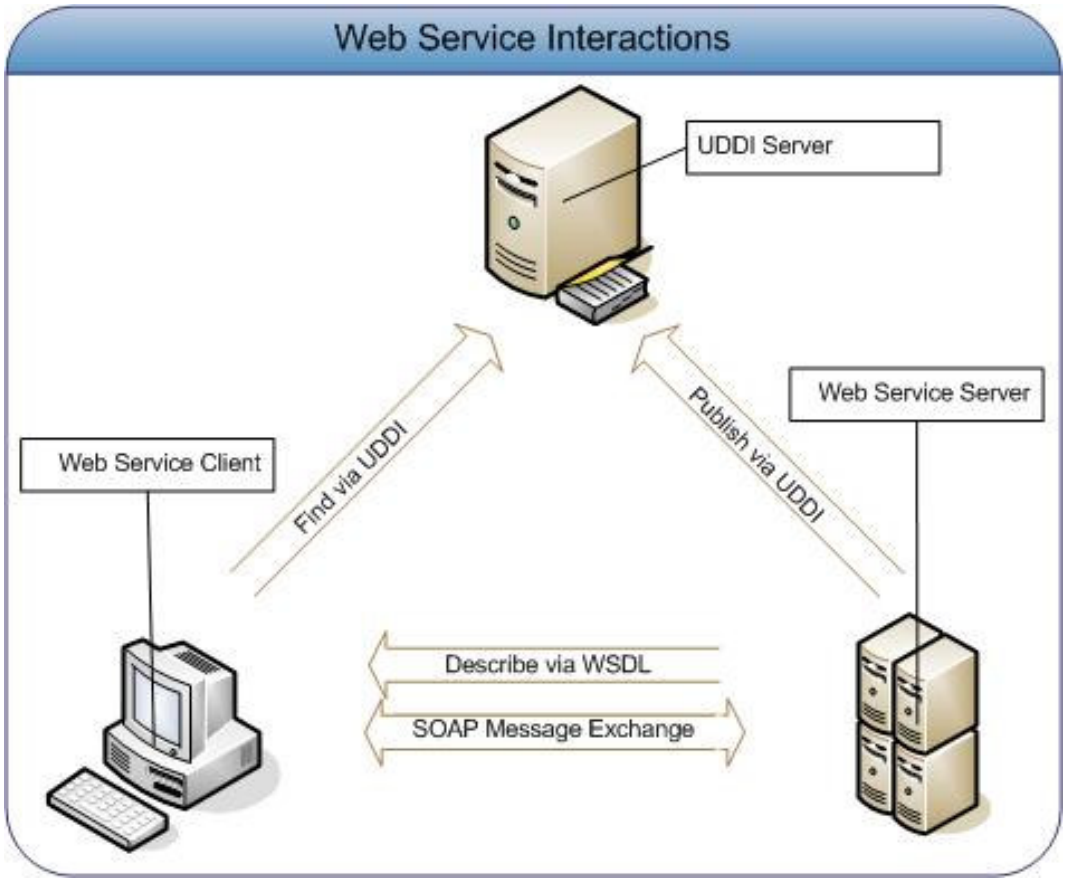

Figure 2.1 – Basic operation of Web Service Entities

These standards have effectively become de facto standards, with effectively universal acceptance and widespread implementation by vendors. However, to improve the security and reliability of Web Services and to address more complex business scenarios, a wide range of additional protocols have since been proposed. Some of these protocols have been standardized, and some are in the process of being standardized. There also some subjects still lacking any common consensus. This stack of technologies is presented in Figure 2.2. A Categorization Web Services standards is as follows;

1. Description and discovery
2. Messaging

3. Management
4. Business Processes
5. Transactions
6. Security

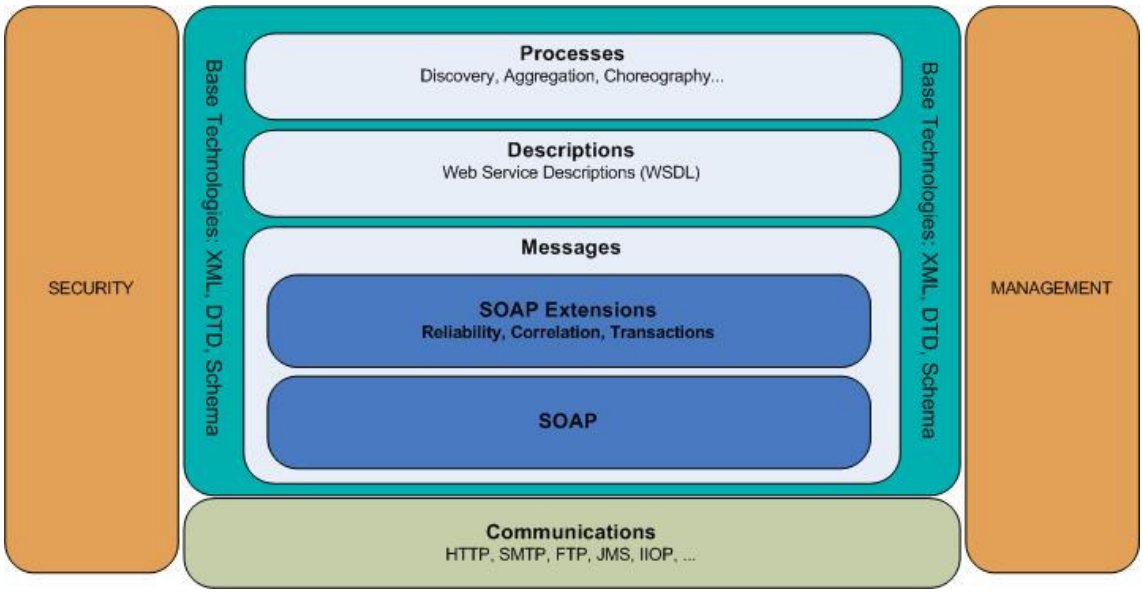

Figure 2.2 Web Services Stack

2.3. Web Services Architectural Models

Web Services Architecture has four basic models. Each model is named after what may be viewed as the key concept of that model. The four models are:

1. The Message Oriented Model
2. The Service Oriented Model
3. The Resource Oriented Model
4. The Policy Model

2.3.1. The Message Oriented Model

The Message Oriented Model focuses on messages, message structure, and message transport without particular reference as to reasons for the messages, nor to their significance. Specifically, in this model, we are not concerned with any semantic significance of the content of a message or its relationship to other messages. However, the Message Oriented Model does focus on the structure of the messages, on the relationship between message sender and receivers and how messages are transmitted.

2.3.2. The Service Oriented Model

The Service Oriented Model focuses on aspects of service, action and so on. While clearly, in any distributed system, services can not be adequately realized without some means of messaging, the converse is not the case: messages do not need to relate to services.

The Service Oriented Model makes use of meta-data which is a key property of Service Oriented Architectures. This meta-data is used to document many aspects of services: from the details of the interface and transport binding to the semantics of the service and what policy restrictions there may be on the service.

2.3.3. The Resource Oriented Model

The Resource Oriented Model focuses on resources that exist and have owners. Resources are a fundamental concept that underpins much of the Web and much of Web services; for example, a Web service is a particular kind of resource that is important to this architecture.

2.3.4. The Policy Model

The Policy Model focuses on constraints on the behavior of agents and services. We generalize this to resources since policies can apply equally to documents (such as descriptions of services) as well as active computational resources.

The Policy Model focuses on those aspects of the architecture that relate to policies and, by extension, security and quality of service. Security is fundamentally about constraints; about constraints on the behavior on action and on accessing resources. Similarly, quality of service is also about constraints on service. In the PM, these constraints are modeled around the core concept of policy; and the relationships with other elements of the architecture. Thus the PM is a framework in which security can be realized.

2.4. Advantages and Disadvantages of the Web Services Technology

Compared to other technologies for building distributed systems, The Web Services technology naturally has advantages and disadvantages. The keen focus on leveraging existing infrastructure during the initial design phase of the technology provides the foundation for its advantages, while its test based nature results in both some of the advantages and disadvantages.

Web Services provide interoperability between various software applications running on disparate platforms/operating systems. Web services use open standards and protocols. Protocols and data formats are text-based where possible, making it easy for developers to comprehend. XML is the agreed upon data format universally used by Web Services implementations. Its text based nature is not actually a requirement as commonly mentioned but rather a choice of serialization mechanism. However, regardless of the serialization mechanism, the data format specified by the XML Infoset is universally the same, thus enabling interoperability across platforms.

Web Services technology utilizes the HTTP protocol, therefore, can work through many common firewall security measures while other forms of RPC may often be blocked. This is an important advantage of the technology from a systems management and security point of view.

Web Services are loosely coupled thereby facilitating a distributed approach to application integration. Loosely Coupled systems are considered useful when either the source or the destination computer systems are subject to frequent changes.

On the other hand, Web Services technology suffers from the fact that it is relatively a new technology. Important standards such as transactions are currently nonexistent or are very immature, compared to mature distributed computing standards

such as CORBA. This is likely to be a temporary issue, since most vendors have committed to the Web Services technology around two main organizational bodies, namely the W3C and OASIS. Lacking standards are in the process of creation.

Another drawback of Web Services might be its poor performance compared to other distributed computing approaches. This is a common problem with text based encoding approaches. Although XML does not consider conciseness or efficiency of parsing among its goals, new standards such as the XML Infoset provides other viable alternatives by describing XML in terms of abstractions. Consequently, binary serialization of XML documents becomes an equally valid alternative. Also SOAP MOTM standard promises to improve the wire efficiency of XML.

2.5. An Evaluation of the Current Architecture

We believe the architecture proposed by the W3C working group substantially meets the requirements for a complete distributed system with the exception of security and privacy. Although there exists substantial material that lays the foundation for addressing these, there is definitely room for more work.

CHAPTER 3

WEB SERVICES SECURITY

3.1 A Security Primer

Since the early days of the Internet as a universal network open to anyone, secure information exchange has been a concern. While it is worth noting that there is no absolute security, without an appropriate level of security, the commercial exploitation of the Internet would not be feasible. While defining a security model, how data flow through an application and over a network to meet the requirements defined by the business without exposing the data to any risk must be shown.

There are seven requirements that must be addressed by a general security framework as defined by the ISO Security Standard:

1. Identification: The party accessing the resource is able to identify itself to the system.
2. Authentication: Authentication is the process of validating the user, whether a client is valid in a particular context. A client can be an end user, a machine or an application.
3. Authorization: Authorization is the process of checking whether the authenticated user has access to the requested resource.
4. Integrity: Ensure that the information will not be changed, altered, or lost in an unauthorized or accidental manner.
5. Confidentiality: No unauthorized party or process can access or disclose the information.
6. Auditing: All transactions are recorded so that problems can be analyzed after the fact.
7. Non-repudiation: Both parties are able to provide legal proof to a third party that the sender did send the information, and the receiver received the identical information. Neither involved side is unable to deny.

Some classifications also include availability to be a part of the schema above schema meaning that hostile attack can not achieve denial-of-service by allocating too many system resources. In this dissertation, we will not deal with this security aspect.

3.2 Web Services Security Exposures

Web Services security is one of the most important Web Services subjects. When using Web Services, similar security exposures exist as for other Internet, middleware-based applications and communications. To demonstrate the Web Services security exposures, we are going to explain several major risk factors for a system with no security. Most common security risks has been depicted in Figure 3.1.

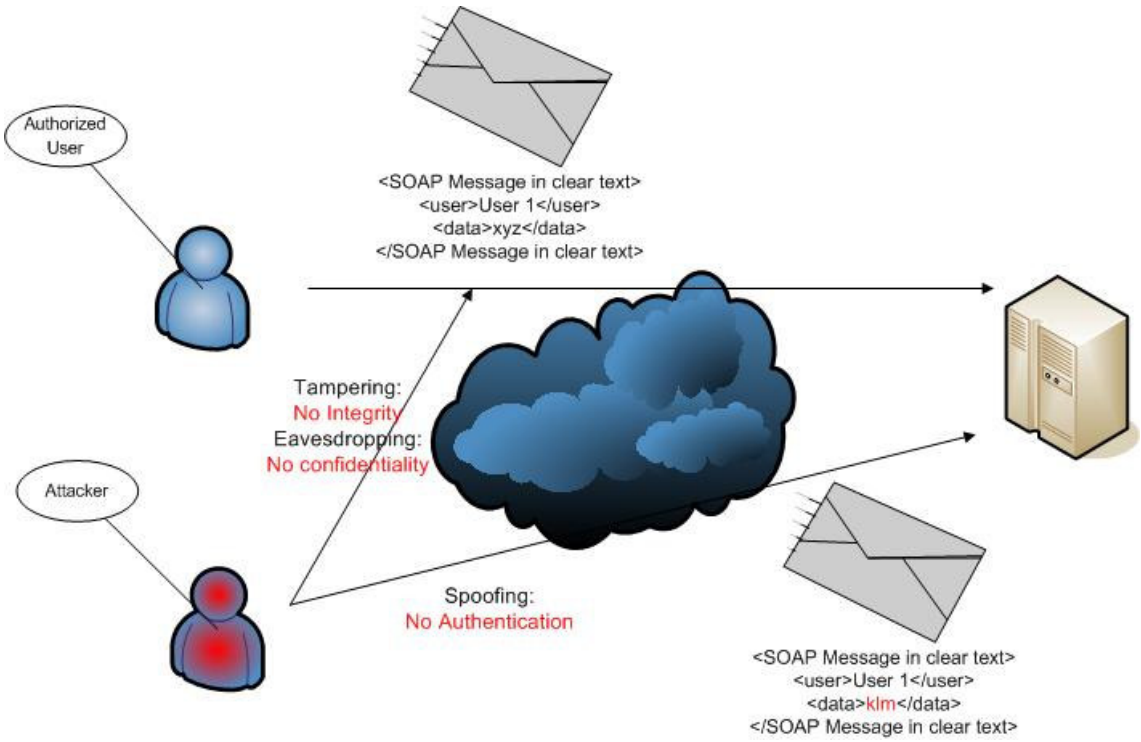

Figure 3.1 Web Services Scenario with No Security

Spoofing – no authentication: An attacker could send a modified SOAP message to the service provider, pretending to be a legitimate user, to get confidential information, or to perform unauthorized acts. By applying authentication to the Web Service, this security exposure can be eliminated.

Tampering – no integrity: The soap message is intercepted between the Web Service client and the server. An attacker could modify the message en-route and since there is no integrity constraint, the Web Service server does not check if the message is

valid and will accept the modified transaction. By applying a confidentiality mechanism to the Web Service, this security exposure can be eliminated.

Eavesdropping – no confidentiality: An attacked could intercept the SOAP message and read all contained information. Because the message is not encrypted, confidential information can be obtained by the attacker. This exposure exists since the message is sent in plain text. By applying a confidentiality mechanism, this security exposure can be eliminated.

3.3. Security Measures for Web Services

When enabling security for any application, the real challenge is in understanding and assessing the risk involved and applying the appropriate measures. For Web Services, knowing what security technology exists today, tracking new emerging standards, and understanding how they will be used to offset the risk is crucial.

3.3.1. General Effect of Security on Information Systems

The more security mechanisms implemented, which increases the security effect, the more influence on other non-functional requirements is given. Therefore, when designing a Web Services security solution, one has to keep in mind that security has an impact on several non-functioning requirements.

3.3.1.1. Security Effect on System Capacity

Any applied security mechanism has an impact on system resource usage, such as processor and memory usage. Therefore, when planning a Web Service environment, the required security overhead must be considered in the capacity and volume planning.

The non-functional requirements, capacity and volume, cover for example the number of concurrent users, and the number of transactions per second. This has influence on the required system infrastructure in terms of hardware and network.

3.3.1.2. Security Effect on Performance

Security mechanisms and functions also impact the applications response time. When defining the Web Service system response time requirements, one has to mind that the response time will be affected when applying the security.

The performance requirement for a system defined the response time for a main application operation (for example: less than 1 second response time for 90% of all transactions).

3.3.2. Web Services Security Approaches

Threats to Web Services involve threats to the host system, the application and the entire network infrastructure. To secure Web Services, a range of XML-based security mechanisms are needed to solve problems related to authentication, role based access control, distributed security policy enforcement, message layer security that accommodate the presence of intermediaries. Web services implementations may require point-to-point and/or end-to-end security mechanisms, depending upon the degree of threat or risk. Traditional, connection-oriented, point-to-point security mechanisms may not meet the end-to-end security requirements of Web services. However, security is a balance of assessed risk and cost of countermeasures. Depending on implementers risk tolerance, point to point transport level security can provide enough security countermeasures.

3.3.2.1. Security Policies

Three fundamental concepts related to Web services security exist from the perspective of Web services architecture: the resources that must be secured, the mechanisms by which these resources are secured, and policies, which are machine-processable documents describing constraints on these resources.

Policies can be logically broken into two main types: permission policies and obligatory policies. A permission policy concerns those actions that an entity is permitted to perform and an obligatory policy concerns those actions that an entity is required to perform. Due to their nature, these two different kinds of policies have

different types of enforcement mechanisms. A permission policy guard mechanism can be used to verify that a requested action is permitted to be performed by the entity, while the obligatory guard mechanism can only verify after the fact that an obligation has not been met. The architecture is principally concerned with the existence of such guard mechanisms and their role in the architecture.

Not all guards are active processes. For example, confidentiality of messages is provided by encryption. The guard here is the encryption itself, although this may be further backed up by active guards that apply policy.

3.3.2.2. Message Level Security

Traditional network level security mechanisms such as Transport Layer Security (SSL/TLS), Virtual Private Networks (VPN), IPSec (Internet Protocol Security), and Secure Multipurpose Internet Mail Exchange (S/MIME) are point-to-point technologies. Although these traditional technologies may be used for Web services security, they are not sufficient for providing an end-to-end security context, ad Web services use a message oriented approach that enables complex interactions that can include the routing of messaged across various trust domains.

Therefore, message level security is important as opposed to point-to-point transport level security. As can be seen in figure 3.2 below, the security context of SOAP message is end-to-end. However, there may also be a need for the intermediary to have access to some information in the message. This is illustrated as a security context between the intermediary and the original requester agent, and the intermediary and the ultimate receiver.

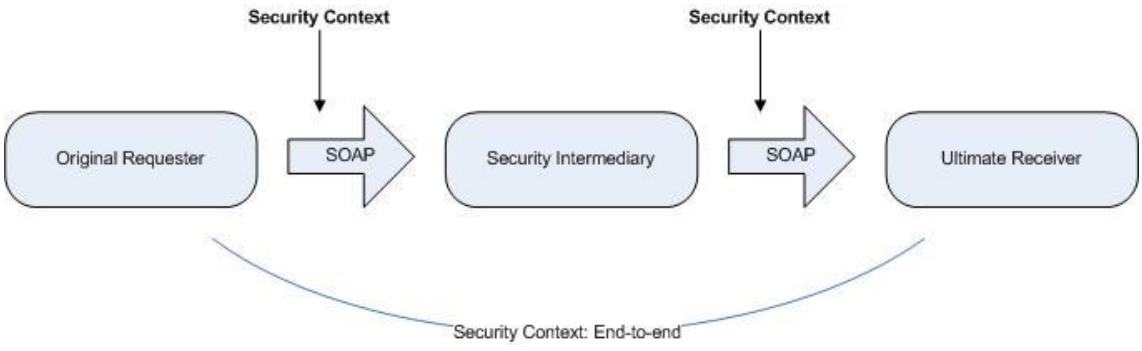

Figure 3.2 Point-to-point vs. End-to-end Security

3.3.2.3. Transport Level Security

HTTP, the most widely used Internet communication protocol, is currently also the most popular protocol for Web services. HTTP is an inherently insecure protocol since all information is sent in clear text between unauthenticated peers over an insecure network. To secure HTTP transport-level security can be used. Transport-level security is a well known and often used mechanism for securing Internet and Intranet communications. It is based on Secure Sockets Layer or Transport Layer Security that works beneath HTTP.

HTTPS allows client and server side authentication through certificates, which have been either self-signed or signed by a central agency. HTTPS can be combined with any part of message level security.

Although HTTPS does not cover all aspects of a general security framework, it provides a security level regarding party identification and authentication, message integrity and confidentiality. SSL can not be applied to other protocols such as JMS. Using SSL point-to-point security can be achieved.

3.3.3. Web Services Security Technologies

In Web services, the SOAP envelope is defined in XML, thus, Web services can use many of the existing XML security technologies and standards, such as XML encryption and XML Digital Signatures. In addition many new standards such as WS-Security have emerged. The WS-Security is the cornerstone of all the efforts in pulling all these requirements together. The abstract of WS-Security specification document says that "WS-Security describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies."[3] Other technologies in the process of standardization are XML Key Management Specification (XKMS), Secure Assertion Markup Language (SAML), Extensible Access Control Markup Language (XACML), and Identity Federation.

3.3.3.1. XML Signature and XML Encryption

XML signatures are designed for use in XML transactions. It is a standard that is jointly developed by W3C and the IETF (RFC 2807 and RFC 3275). The standard defines a schema for capturing the result of a digital signature operation applied to arbitrary data and its processing. XML signatures add authentication, data integrity, and support for non-repudiation to the signed data.

XML Encryption specifies a process for encrypting the data and representing the result in XML. The data may be arbitrary data (including an XML document), an XML element, or XML element content. The result of encrypting data is an XML Encryption element which contains or references the cipher data.

3.3.3.2. Web Services Security

The WS-Security specification provides message-level security, which is used when building secure Web services to implement message content integrity and confidentiality. The advantage of using WS-Security over using SSL is that WS-Security can provide message level end-to-end security. This means that security can be maintained even if the message goes through multiple services, called intermediaries. Additionally, WS-Security is independent of the transport layer protocol meaning that it can be used for any Web service binding such as HTTP or RMI.

The WS-Security specification defines a set of SOAP extensions. The specification is flexible and is designed to be used as the basis for securing Web services in a wide variety of security models, including PKI, Kerberos and SSL. It provides support for multiple security token formats, multiple trust domains, multiple signature formats, and multiple encryption technologies based on XML signature and XML encryption to provide integrity and confidentiality.

The specification includes security token propagation, message integrity, and message confidentiality. However, these mechanisms by themselves do not address all the aspects of a complete security solution, therefore, WS-Security represents only one of the layers in a complex, secure Web services security solution design.

Message integrity is provided by leveraging XML signature in conjunction with security tokens (which may contain or imply key data) to ensure that messages are

transmitted without modifications. The integrity mechanisms are designed to support multiple signatures potentially by multiple actors, and to be extensible to support additional signature formats. The signatures may reference a security token.

Similarly, message confidentiality is provided by leveraging XML Encryption in conjunction with security tokens to keep portions of SOAP messages confidential. The encryption mechanisms are designed to support additional encryption technologies, processes, and operations by multiple actors. The encryption might also reference a security token "(IBM CORPORATION, 2002)".

3.3.3.2.1. Advantages of WS-Security

Multiple parts of a message can be secured in different ways. Multiple security requirements can be applied, such as integrity on the security token, and confidentiality on the SOAP body. End-to-end message level security can be provided through any number of intermediaries. WS-Security works across multiple transports and is independent of the underlying transport. Authentication of multiple party identities is possible.

3.3.3.2.2. Evolution of WS-Security Specification

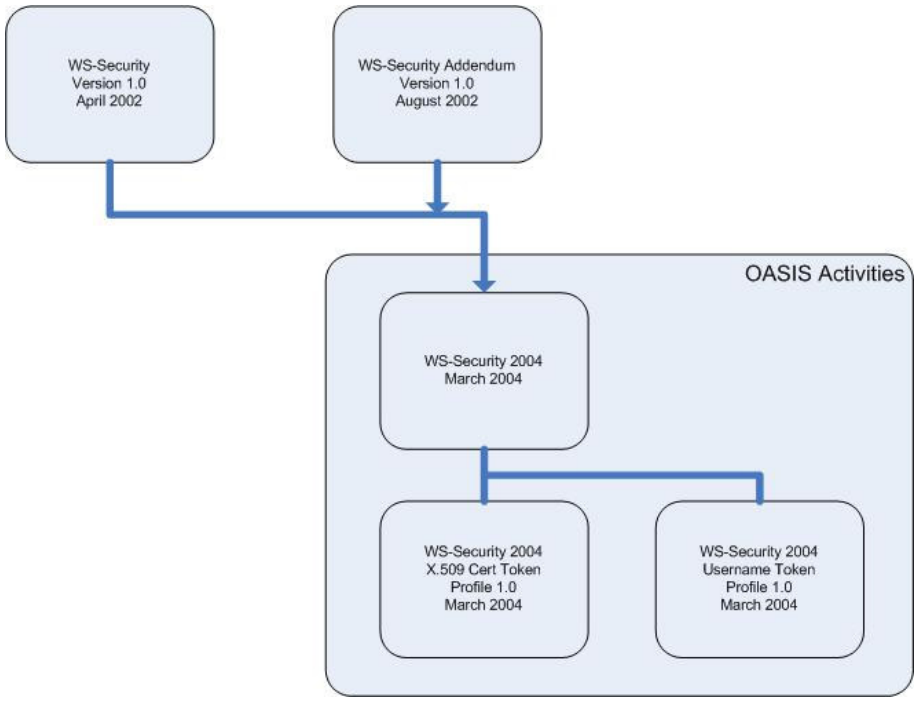

Figure 3.3 Evolution of WS-Security Specification

The first version of WS-Security specification was proposed by IBM, Microsoft, and Verisign in April 2002. After the formalization of the April 2002 specifications, the specification was transferred to OASIS consortium.

In OASIS activities, the core specification and many profiles that describe the use of a specific token framework in WS-Security have been discussed. The latest specification and profiles of WS-Security were proposed in March 2004 as an OASIS standard.

The latest core specification, Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) was standardized in March 2004. The two profiles, Web Services Security Username Token Profile, and Web Services Security X.509 Certificate Token Profile 1.0 were standardized at the same time. This evolution is presented in Figure 3.3.

There are other token profiles on which OASIS is currently working on:

- Web Services Security: SAML Token Profile
- Web Services Security: Rights Expression Language (REL) Token Profile
- Web Services Security: Kerberos Token Profile
- Web Services Security: Minimalist Profile (MProf)
- Web Services Security: SOAP Message with Attachments (SWA) Profile

3.3.3.2.3. Web Services Security Model Framework

The WS-Security specification addresses only a subset of security services for all security aspects. A more general security model is required to cover other security aspects such as auditing and non-repudiation.

The Web services security model introduces a set of individual interrelated specifications to form a layering approach to security. This layered architecture is presented in Figure 3.4. It includes several aspects of security: identification, authentication, authorization, integrity, confidentiality, auditing, and non-repudiation.

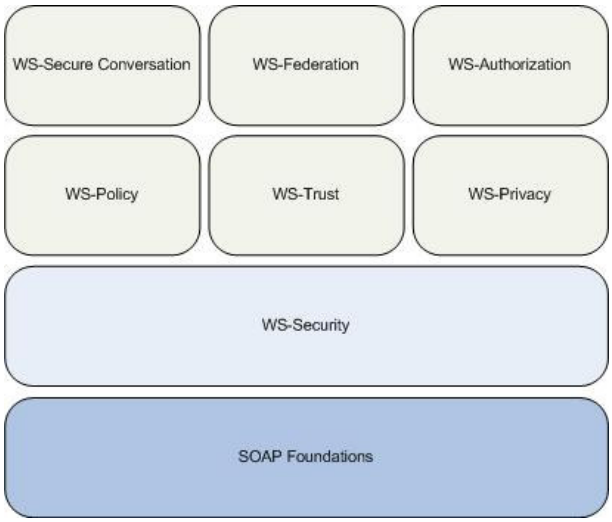

Figure 3.4 Web Services Security Specifications

These specifications include different aspects of Web services security:

WS-Policy – Describes the capabilities and constraints of the security (and other business) policies on intermediaries and endpoints (e.g. required security tokens, supported encryption algorithms, privacy rules).

WS-Trust – Describes a model for trust frameworks that enables Web services to securely interoperate.

WS-Privacy – Describes a model for how Web services and requesters state subject privacy preferences and organizational privacy practice statements.

WS-SecureConversation – Describes how to manage and authenticate message exchanges between parties including security context exchange and establishing and deriving session keys.

WS-Federation – Describes how to manage and broker the trust relations in a heterogeneous federated environment including support for federated identities.

WS-Authorization – Describes how to manage authorization and data and authorization policies.

3.3.3.2.4. WS-Security Example

This section provides examples of SOAP messages with WS-Security. Using WS-Security, the authentication mechanism integrity, and confidentiality can be applied

at the message level. As an overview, Figure 3.5 shows an example of Web service security elements when the SOAP body is signed and encrypted.

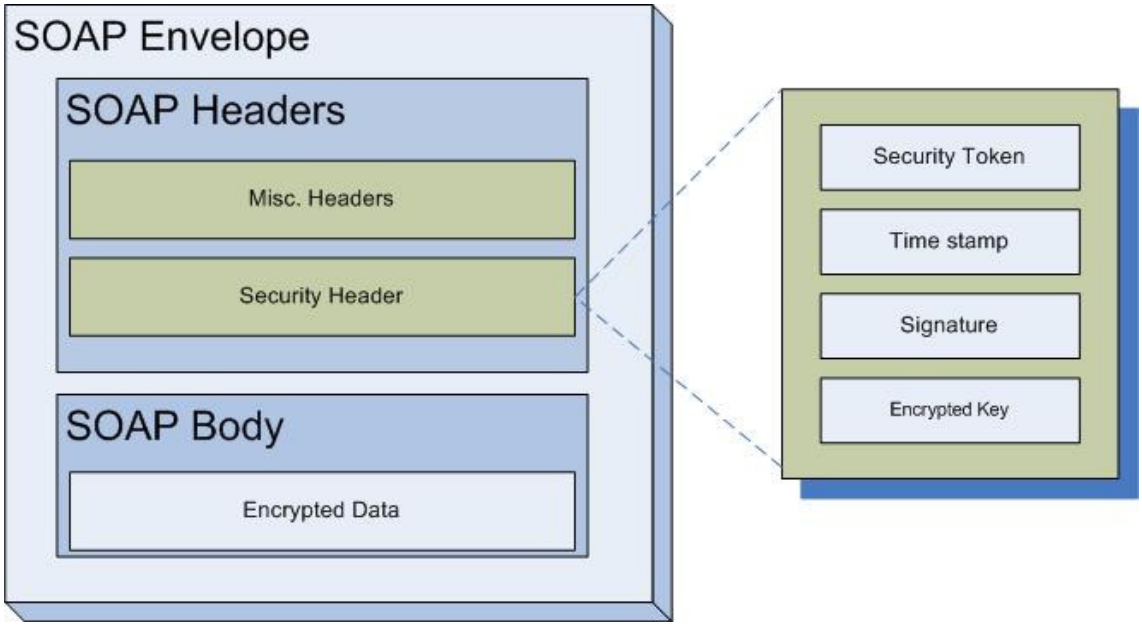

Figure 3.5 SOAP Message Security with WS-Security

Example 3.1 shows the sample SOAP message without applying WS-Security. As you can see, there is only a SOAP body under the SOAP envelope. Applying WS-Security, the SOAP security header will be inserted under the SOAP envelope.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Header/>
    <soapenv:Body>
    <p821:getDayForecast xmlns:p821="http://bean.itso">
        <theDate>2004-11-25T15:00:00.000Z</theDate>
    </p821:getDayForecast>
    </soapenv:Body>
    </soapenv:Envelope>
```

Example 3.1 SOAP message without WS-Security

In the sections that follow, we will show examples with WS-Security applied to the SOAP message.

3.3.3.2.4.1. Authentication

In example 3.2, we show a message with authentication. As can be seen, we have a username and password information as a <UsernameToken> tag in the message "(OASIS, 2006)". When the username token is received by the Web service server, the username and password are extracted and verified. Only when the username and password combination is valid will the message be accepted and processed at the server.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Header>
      <wsse:Security soapenv:mustUnderstand="1"
          xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401 -
wsswssecurity- secext-1.0.xsd">
        <wsse:UsernameToken>
        <wsse:Username>David</wsse:Username>
        <wsse:Password
                Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wssusername-
                 token-profile-1.0#PasswordText">divaD</wsse:Password>
        </wsse:UsernameToken>
      </wsse:Security>
    </soapenv:Header>
    <soapenv:Body>
    <p821:getDayForecast xmlns:p821="http://bean.itso">
        <theDate>2004-11-25T15:00:00.000Z</theDate>
    </p821:getDayForecast>
    </soapenv:Body>
    </soapenv:Envelope>
```

Example 3.2 SOAP Message with Authentication

Using the username token is just one of the ways of implementing authentication. This mechanism is also known as basic authentication. Any XML based security token can be specified in the <Security> header. However, binary tokens such as X.509 certificates, and Kerberos tickets, or other non-XML formats require a special encoding format for inclusion.

3.3.3.2.4.2. Integrity

Integrity is applied to the application to ensure that no one illegally modifies the message while it is in transit. Essentially, integrity is provided by generating an XML digital signature on the contents of the SOAP message. If the message data changes illegally, the signature would no longer be valid.

Example 3.3 shows a sample SOAP message with integrity. Here the message body part is signed and added to the SOAP security header as signature information.

```
<saml:Assertion AssertionID="SecurityToken-07dc7c16-9a42-4100-ad21-
13013b975f3c"
MajorVersion="1" MinorVersion="1"
Issuer="http://localhost/SamlSecurityTokenService/SamlTokenIssuer.ashx"
IssueInstant="2005-11-29T22:36:03Z"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
<saml:Conditions NotBefore="2005-11-29T22:36:03Z" NotOnOrAfter="2005-11-
30T02:36:03Z">
<saml:AudienceRestrictionCondition>
<saml:Audience>
http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
</saml:Audience>
</saml:AudienceRestrictionCondition>
</saml:Conditions>
<saml:AuthenticationStatement
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
AuthenticationInstant="2005-11-29T22:36:03Z">
<saml:Subject>
<saml:NameIdentifier
Format="http://schemas.xmlsoap.org/ws/2004/01/Federation/username">RDALAPTOP
02\wse
user</saml:NameIdentifier>
<saml:SubjectConfirmation>
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-ofkey</
saml:ConfirmationMethod>
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<xenc:EncryptedKey
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
<xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
<wsse:SecurityTokenReference>
```

Example 3.3 SOAP Message with Integrity

```
<wsse:KeyIdentifier ValueType="http://docs.oasisopen.
org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1"
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soapmessage-
security-
1.0#Base64Binary">aAI1zTqHbhsUN6j2HsIefWcHODs=</wsse:KeyIdentifier>
</wsse:SecurityTokenReference>
</KeyInfo>
<xenc:CipherData>
<xenc:CipherValue>cSrRHLh8DWeELh5Naf34z515OIs0sw6hp4/zUIOnbOFNj1lxQd
XCTi7z3aaLu4Xi
ws8vF3YdzD9LD/bQ1+QzzI7qcR4eDLNnxjZU87DkCBxI4ygqyB+Mx4J2lKLYl+rxI
OVOVcjbd64/YngQu5
AgZKBxNZv7GIcla0d3Ikebyr4=</xenc:CipherValue>
</xenc:CipherData>
</xenc:EncryptedKey>
</ds:KeyInfo>
</saml:SubjectConfirmation>
</saml:Subject>
<saml:SubjectLocality IPAddress="192.168.0.10" DNSAddress="CLIENTHOST" />
</saml:AuthenticationStatement>
<saml:AttributeStatement>
<saml:Subject>
<saml:NameIdentifier
Format="http://schemas.xmlsoap.org/ws/2004/01/Federation/username">RDALAPTOP
02\wse
user</saml:NameIdentifier>
<saml:SubjectConfirmation>
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-ofkey</
saml:ConfirmationMethod>
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<xenc:EncryptedKey
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
<xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
<wsse:SecurityTokenReference>
<wsse:KeyIdentifier ValueType="http://docs.oasisopen.
org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1"
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soapmessage-
security-
1.0#Base64Binary">aAI1zTqHbhsUN6j2HsIefWcHODs=</wsse:KeyIdentifier>
</wsse:SecurityTokenReference>
</KeyInfo>
<xenc:CipherData>
```

Example 3.3 (Continued)

```
<xenc:CipherValue>cSrRHLh8DWeELh5Naf34z515OIs0sw6hp4/zUIOnbOFNj1lxQd
XCTi7z3aaLu4Xi
ws8vF3YdzD9LD/bQ1+QzzI7qcR4eDLNnxjZU87DkCBxI4ygqyB+Mx4J2lKLYl+rxI
OVOVcjbd64/YngQu5
AgZKBxNZv7GIcla0d3Ikebyr4=</xenc:CipherValue>
</xenc:CipherData>
</xenc:EncryptedKey>
</ds:KeyInfo>
</saml:SubjectConfirmation>
</saml:Subject>
<saml:Attribute AttributeName="group"
AttributeNamespace="http://schemas.xmlsoap.org/ws/2004/01/Federation/group">
<saml:AttributeValue>BUILTIN\Users</saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-excc14n#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"
/>
<Reference URI="#SecurityToken-07dc7c16-9a42-4100-ad21-13013b975f3c">
<Transforms>
<Transform Algorithm="http://www.w3.org/2000/09/xmldsig#envelopedsignature"
/>
<Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
</Transforms>
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
<DigestValue>k2PBIDmwJLlQIh/GA4bVPgk1544=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>CZPQJvc2vpjKyio6EEJTQShHGlJff1hZubOWSDCbRrbSFkK53fOp
9Ugulfel6vvs9y
ZfFJB2ieRAPK3ywUrRWTKvO7vjXP9HRGgrvqGC2PpQNAEYn7ciBkLM+VoJV5v
WfIopVevEYvnxFMFZlJTl
LSOr0n+GWYaZuUYFJECfHaE=</SignatureValue>
<KeyInfo>
<wsse:SecurityTokenReference>
<wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasiswss-
soap-message-security-1.1#ThumbprintSHA1" EncodingType="http://docs.oasisopen.
org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary">vLfjdZyxqw+ztcYkVVQX8wyXp5o=</wsse:KeyIdentifier>
</wsse:SecurityTokenReference>
</KeyInfo>
</Signature>
</saml:Assertion>
```

Example 3.3 (Continued)

A signature is based on a key that the sender is authorized to have. Unauthorized sniffers do not have this key. When the receiver gets the message, it too creates a signature using message contents. Only if the two signatures match does the receiver honor the message. If the signatures are different, a SOAP fault is returned to the sender.

3.3.3.2.4.3. Confidentiality

Example 3.4 shows a sample SOAP message with confidentiality. Here, the message body part is encrypted and a security header with encryption information added. Confidentiality is the process in which a SOAP message is protected so that only authorized recipients can read the SOAP message. Confidentiality is provided by encrypting the contents of the SOAP message using XML encryption. If the SOAP message is encrypted, only a service that knows the key can decrypt and read the message.

```
<soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Header>
        <wsu:Timestamp
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-ws
            security-utility-1.0.xsd">
            <wsu:Created>2004-11-26T09:34:50.838Z</wsu:Created>
        </wsu:Timestamp>
        <wsse:Security soapenv:mustUnderstand="1"
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-w
            ssecurity-secext-1.0.xsd">
            <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
                <EncryptionMethod
                    Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
                    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                        <wsse:SecurityTokenReference>
                        <wsse:KeyIdentifier
                        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
                        200401-wss-x509-token-profile-
1.0#X509v3SubjectKeyIdentifier">
Vniy7MUOXBumPoH1MNbDpiIWOPA=
                        </wsse:KeyIdentifier>
```

Example 3.4 SOAP Message with Confidentiality

```
                        </wsse:SecurityTokenReference>
                    </ds:KeyInfo>
                <CipherData>
                    <CipherValue>
O+2mTsRjU1iNlwANv1kGdzpkRV1GQc5epAT3p5Eg5UNAJ3H3YAX5VrdgMQmj1
wzdSZLDEzBtcHPJq3c8c0AgmAy9EVdcgXIn/ZeV+80jMDn/HN2HfodYjURtIYBg48
0SSkot0fy+YpBSXNR/MTfs1HT2H/Mjw/CyIbomWdQZHmE=
                    </CipherValue>
                </CipherData>
                <ReferenceList>

<DataReferenceURI="#wssecurity_encryption_id_6866950837840688804"/>
                </ReferenceList>
            </EncryptedKey>
        </wsse:Security>
    </soapenv:Header>
    <soapenv:Body>
        <EncryptedData
            Id="wssecurity_encryption_id_6866950837840688804"
            Type="http://www.w3.org/2001/04/xmlenc#Content"
            xmlns="http://www.w3.org/2001/04/xmlenc#">
            <EncryptionMethod
                Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
            <CipherData>
                <CipherValue>
OvLekOlbuZhFBllBNL4Kos195YHwYw0kSbMxkbI2pk7nl17g0prPS2Ba2hyrXHABG
QVmosWpgqt+zijCPHUQCMwmm3qgFraK11DPMmwP94HvgxlgBmPw1Unt+WM4
aKLNrHDnwwcQX5RO7KT+fhFp4wxFEABwfHqzvTGNK3xRwJE=
                </CipherValue>
            </CipherData>
        </EncryptedData>
    </soapenv:Body>
</soapenv:Envelope>
```

Example 3.4 (Continued)

3.3.3.2.5. Scenarios

In this section, we present several scenarios that can be enabled using the mentioned Web Service security specifications. These scenarios can be commonly encountered in the industry, and present an opportunity for elaborating on the specifications.

Note that in the scenario descriptions the use of the term requester is used to describe the broad variety potential users of a Web service and is not meant to limit the

characteristics of the requester. In scenario figures, the blue boxes represent the service and the light blue boxes represent security tokens and their identity and delegation claims.

3.3.3.2.5.1. Direct Trust Using Username/Password and Transport-Level Security

This is a very basic example showing how Web services Security can be used with existing security mechanisms.

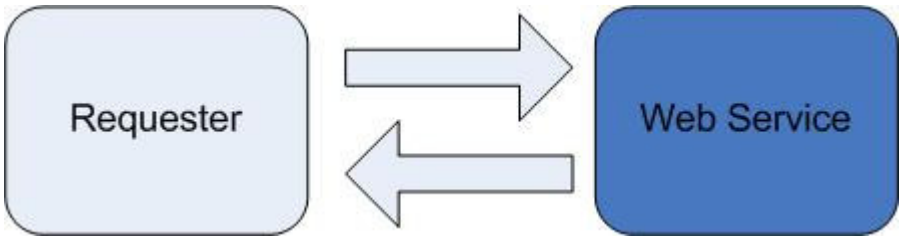

Figure 3.6 Web Services Security and Existing Transport Security Mechanisms

The requester opens a connection to the Web Service using a secure transport such as SSL or TLS. It sends its request along with a security token that contains its username and password. The service authenticates the requester, processes the request and returns the result. In this scenario, the message confidentiality and integrity are handled using existing transport security mechanisms "(IBM CORPORATION, 2006)".

3.3.3.2.5.2. Direct Trust Using Security Tokens

This scenario illustrates the use of a security token that is directly trusted by a Web service. Here direct trust means that the requester's security token or its signing authority is known and trusted by the Web service. This scenario assumes that the two parties have used some mechanism to establish a trust relationship for use of the security token. This trust may be established manually, by configuring the application, or by using secure transport to exchange keys.

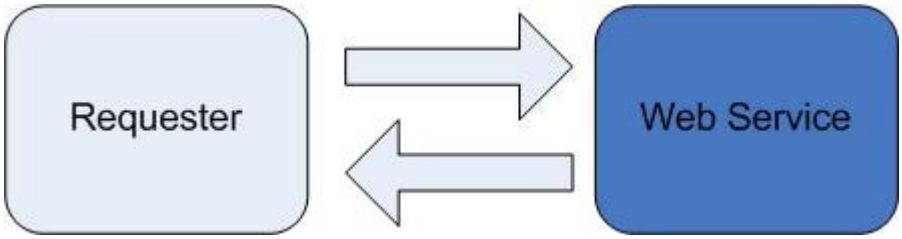

Figure 3.7 Direct Trust between Two Parties

The requester sends a message to a service and includes a signed security token and provides proof of possession of the security token using, for example, a signature. The service verifies the proof and evaluates the security token. The signature on the security token is valid and is directly trusted by the service. The service processes the request and returns a result. Direct trust assumes that the policies for privacy are well understood by the parties involved "(IBM CORPORATION, 2006)".

3.3.3.2.5.3 Security Token Acquisition

In some cases, the security token used is not passed as part of the message, and instead a security token reference is provided that can be used to locate and acquire the token.

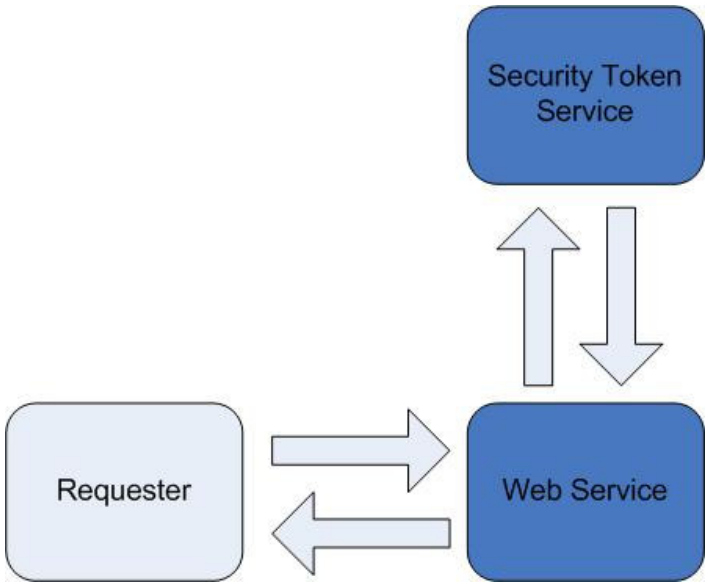

Figure 3.8 Security Tokens by Reference

The requester issues a request to the service and includes a reference to the security token and provides proof-of-possession. The Web service uses the provided information to obtain the security token from the token store service and validate the proof. The Web service trusts the security token, so the request is processed and the response is returned "(IBM CORPORATION, 2006)".

3.3.3.2.5.4. Firewall Processing

Firewalls are a critical component of the current Web security architectures, therefore they should be able to continue enforcing boundary processing rules.

Figure 3.9 below shows the firewall allowing authorized SOAP messages while blocking unauthorized requests.

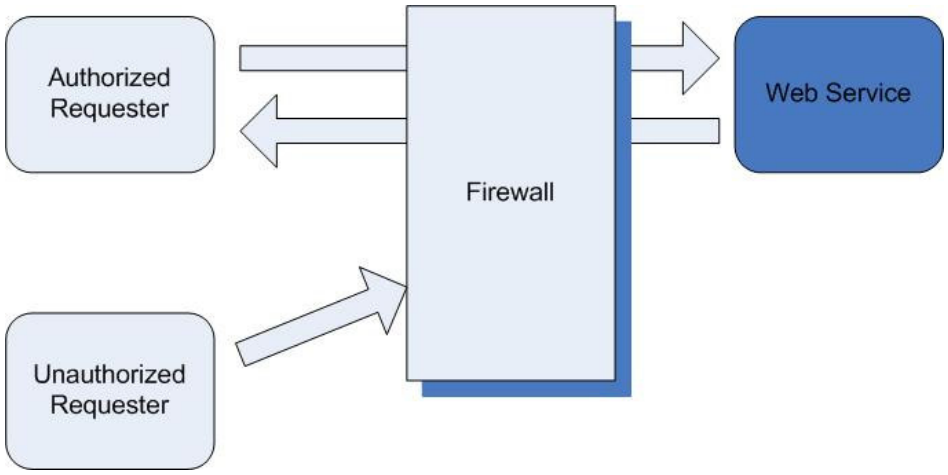

Figure 3.9 Firewall Processing SOAP Messages

In this scenario the firewall examines the SOAP messages to determine if the requests is authorized to send messages to the Web service behind itself. It determines by examining the security token used to sign the message. If the signature is valid, the signing authority is trusted to, and the token says that it does authorize message inside the firewall, then the message is allowed. In other scenarios, the firewall can act as a security token issuing authority and only allow messages that include proof-of-possession of a security token issued by the firewall "(IBM CORPORATION, 2006)".

3.3.3.2.5.5. Issued Security Token

In the first two steps, the requester communicates with a certifying authority to obtain a signed statement of assertions attesting the requester's identity. Next the requester sends the message with the security token and a proof of possession attached to the message. To obtain an identity security token, requesters may use existing security protocols or they may leverage the Web services security specifications "(IBM CORPORATION, 2006)".

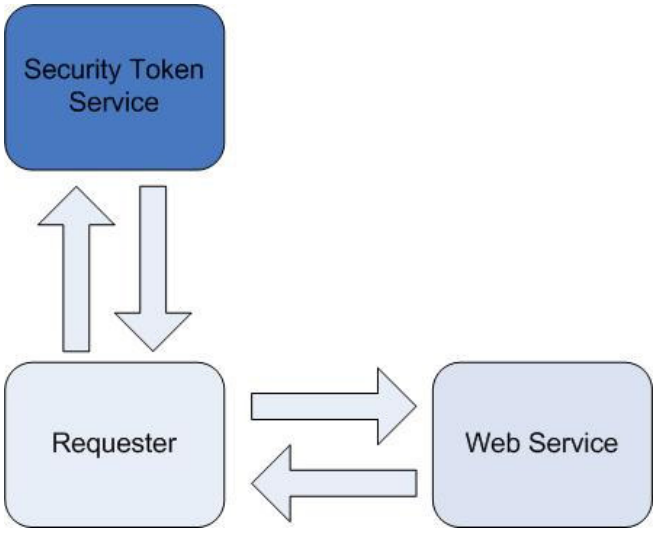

Figure 3.10 Simple Authentication by a Trusted Third Party

3.3.3.2.5.6. Enforcing Business Policy

In many business processes there are specific policies that must be enforced. For example, a service may require that consumers have a certain rating or recognition with a specific external party. With web services, these policies can be codified and validated automatically, simplifying the overall process.

In this scenario, the Web service enables the service provider to interact with partners in order to complete business processes. However, the service provider not only wants to assure of the identity of the requester, but they also want to make sure that the requester has the external party's recognition. The requester would go to the external party and provide its identity security token received from the security token provider in order to prove that it is recognized by the external party. The requester then would go to

the Web service and provide both tokens and prove its identity and that it is recognized and has good standing with the external party "(IBM CORPORATION, 2006)".

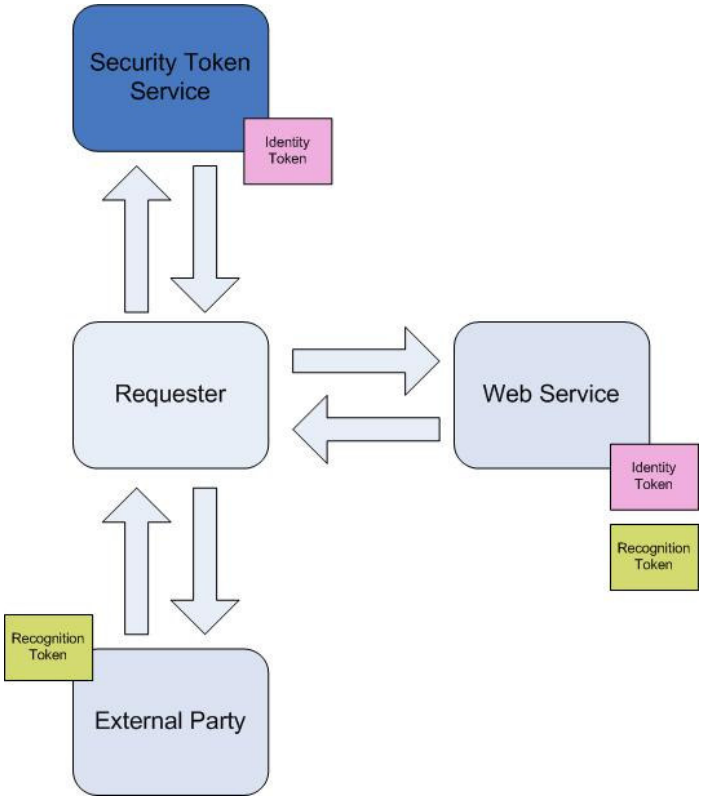

Figure 3.11 An Example in Enforcing Policies

3.3.3.2.5.7. Web Client Communicating Through a Middle-Tier Application to a Service

Consider an example where we have a web client which communicates with a middle-tier web application which, in turn securely talks to a Web service in another domain. The middle-tier web application, which is Web service aware wants to obtain a security token for the Web client in order to talk to the Web service on behalf of the Web client.

To enable this scenario, when the Web client accesses the middle-tier application it is redirected to an associated identity service for authentication possibly by basic HTTP authentication mechanism and transport-layer security. Once authenticated, the request is redirected back to the middle-tier application. The identity

service provides the middle-tier application with a security token asserting the identity and the delegations possibly by a query string sent via https. The Web server can now use this token to issue requests to the Web service. The Web service processes the requests and returns the results to the Web service for possible formatting for display in a browser "(IBM CORPORATION, 2006)".

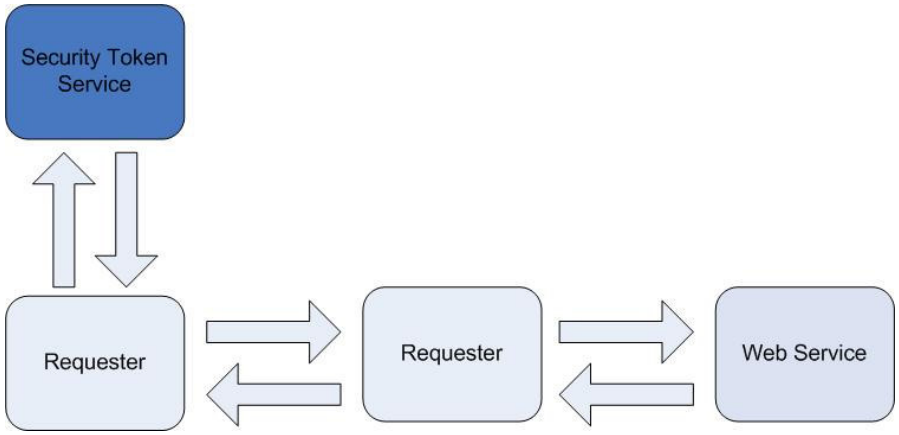

Figure 3.12. Web Client Communicating Through a Middle-Tier Application to a Service

3.3.3.2.5.8. Mobile Clients

The flexibility of the Web services approach enables support for multiple cryptographic technologies providing both strong and performant cryptographic protection on devices with limited computational and storage capabilities. Similarly it enables network operators to provide security proxies, such as network gateways, to act on client's behalf.

When a network operator supports mobile clients using Web services security specifications, they can configure those clients to send requests via the network operator's gateway. In this scenario the gateway is a SOAP intermediary that actively participates in the overall message flow; specifically the network operator is providing a value-add encryption algorithm designed for mobile devices. The gateway can augment or change the security tokens and quality of protection of the message. Note that the flexibility inherent in this Web services security model allows this solution even when the device is roaming on a foreign network "(IBM CORPORATION, 2006)".

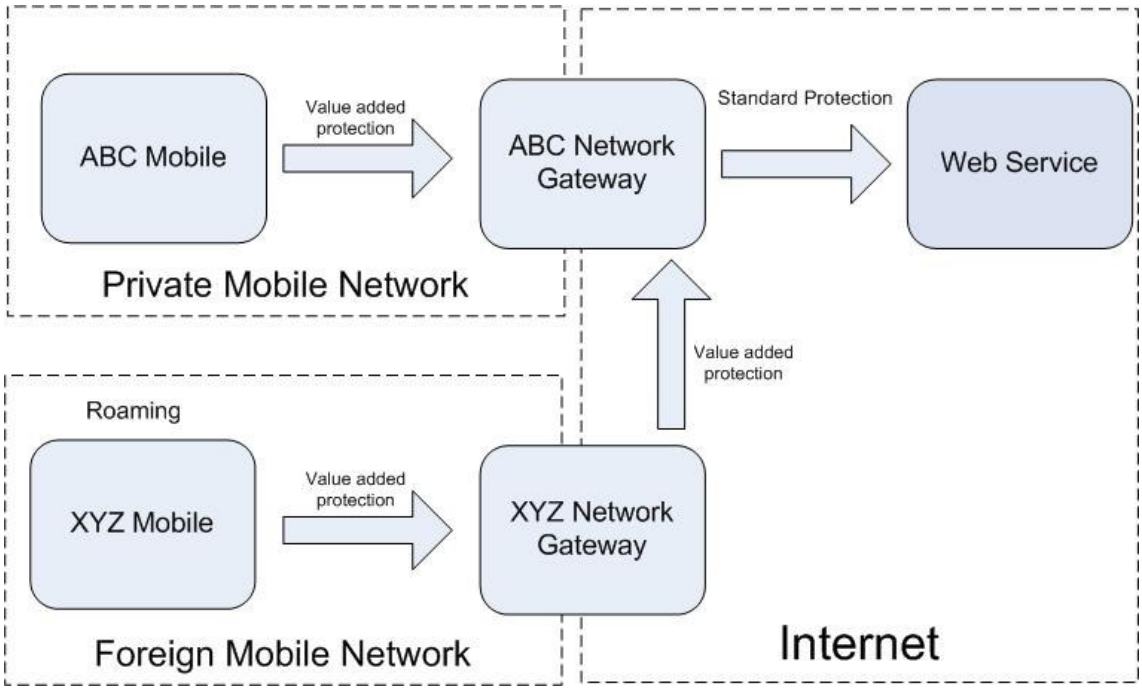

Figure 3.13 Mobile Clients Accessing a Service Through Gateways

CHAPTER 4

CASE STUDY

4.1. Problem Definition

Web services typically have large and dynamic requester populations. While some services allow anonymous access, others require strong user authentication. Invocation of web services commonly occurs between parties between which there is no prior relationship and no common security domain. Daily life provides various examples in which a trust relationship must be formed prior to performance of tasks. Some examples include, request for verification of identity by a bank prior to opening an account, verification of citizenship by government before issuing a passport, and verification of student status by a university library before lending a book.

Web services operate similarly. If a person would like to make an online application for credit to a bank, he should provide a security token proving his identity. A more advanced scenario would be supplying credit status obtained from a governmental institution which monitors citizen's credit rating, along with the token-of-proof of a person's identity. Such a scenario represents the need for a common protocol for representing a person's identity along with various security declarations. Such a security token must be standards based and should be verifiable by different authorities in different security domains.

We propose such an architecture enabling such scenarios in the following sections of this chapter. Our proposed architecture introduces a concept which is called Security Token Service which issues security tokens on demand to requesters. Such tokens may be used for proving identity to other Security Token Services for receiving additional security tokens proving other information about the requester thus should be valid across different security domains.

4.2. Real Life Scenario

The scenario we're going to use for demonstration of our proposed architecture is a real-life example of the WS-Security scenario we have explained in section 3.3.3.2.5.6. Our requester is a citizen who would like to apply for a credit account to a bank. For his application to be honored, the bank requests the applicant to present two security tokens, one obtained from the government verifying his identity, and another token from the fictious agency of Board of Credit Rating which rates people's credit status. When both tokens are acquired, the requester invokes the banks Web service and presents these tokens. The bank verifies these tokens and according to the credit rating of the applicant, and decides to approve the credit request or not.

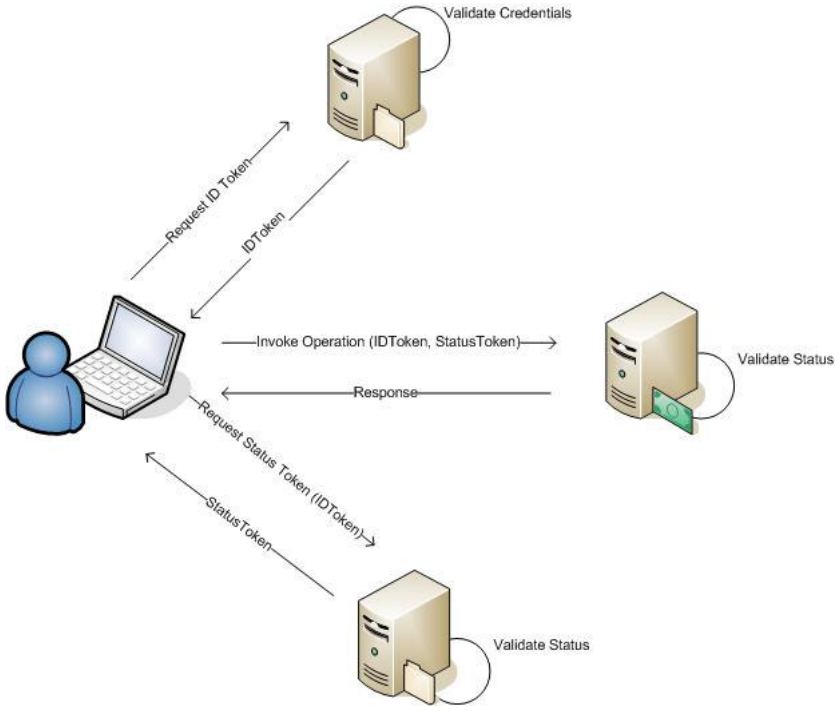

Figure 4.1 Credit Application Scenario

4.3. Motivation

All the actors of the scenario might be implemented using different technologies, and interoperability is required between these platforms. Using a standards based mechanism for authentication helps ensure interoperability between

different platforms. There is also a need for security tokens that are extensible and include claims that support additional security functions. The credit rating of the User is an important piece of information for the Bank Web Service. Using this piece of information, the service decides to honor the User's credit request. Thus, this piece of information is part of the client authorization process on the banking system.

The environment includes organizational boundaries that are protected by firewalls. We need security tokens that can traverse these boundaries, including passing through ports that are commonly enables on firewalls. Also, clients must be able to obtain security tokens such that services in a different security domain can be accessed by the token that is issues by the token issuer in its own domain.

4.4. Architecture

The proposed architecture for this scenario involves 4 parties. The first party is the requestor which is one of the principal actors of the scenario. The requestor can be any kind of application designed to invoke operations on Web services. All of the other parties are Web services. All of the parties are located in a different security domain.

The communication between parties is performed over HTTP, thus avoiding any possible problems regarding firewalls. Our choice of Web service implementation utilizes the classic approach of SOAP over HTTP. We use WS-Security in order to provide integrity and confidentiality. We used WS-Security SAML token profile for interoperability between parties with different security requirements. The last component and the heart of the architecture is WS-Trust technology, which enables us to build Security Token Services that issue on-demand security tokens that can be used for authentication and assertion proof on different domains.

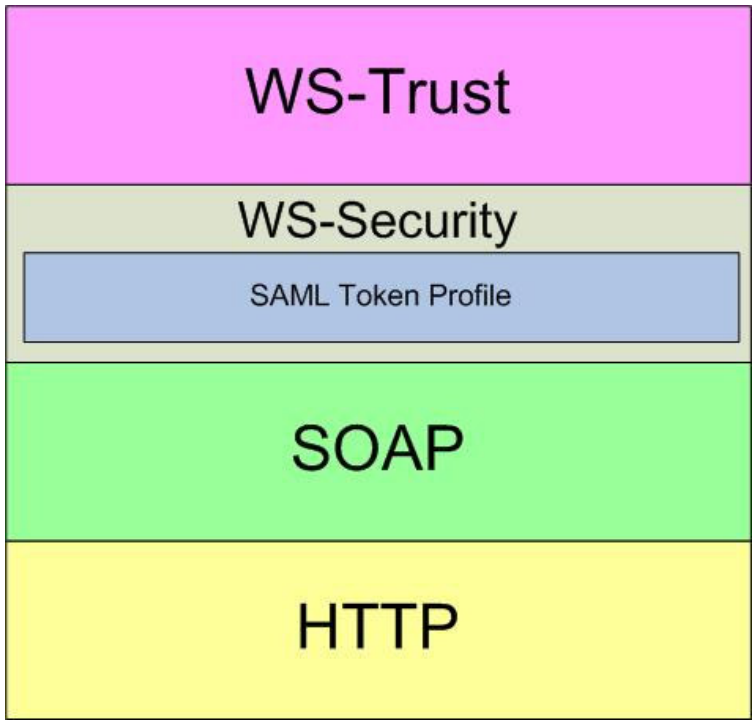

Figure 4.2 The Protocol Stack of the Architecture

4.4.1. Participants

Our scenario involves the following participants:

• Client: The client application that provides the user's credentials for authentication to obtain a SAML token and presents the SAML token in the request to the Web service.

• Security Token Service (STS): The STS is the Web service that authenticates clients by validating credentials that are presented by a client. The STS issues a security token to a client for successful client authentication.

• Service: The service is the Web service that requires authentication of a client prior to authorizing the client.

• User: The authenticated user represented in the issued SAML token. In the SAML token, the authenticated party is referred to as the subject.

In our scenario there are exactly two Security Token Service's. The first one STS accepts User credentials and issues the security token which identifies the user to other parties, and the second one accepts the identity security token, and issues the credit rating security token.

4.4.2. SAML Token Profile

SAML (Security Assertion Markup Language) tokens are standards based XML tokens that are used to exchange security information, including attribute statements, authentication decision statements, and authorization decision statements "(OASIS, 2006)". SAML tokens are also extensible, meaning that the schema of the token can be extended to meet additional requirements. SAML tokens are important for Web service security. They provide cross-platform interoperability, and a means for exchanging security information between clients and services that do not reside in the same security domain.

4.4.3. Web Services Trust

Web Services Trust (WS-Trust) is an extension of the Web Services Security specification. Web Services Trust Language defines extensions of the WS-Security that provides methods for issuing and exchanging security tokens. It also defines a way to establish a trust model "(ANDERSON 2006)".

In order to secure communication between two parties, the two parties must exchange security credentials. However, each party needs to determine if they can trust the asserted credentials of the other party. The WS-Trust specification defines extensions to WS-Security for issuing, exchanging, and validating security tokens, and ways to establish and access the presence of trust relationships "(WEB_3 2006)".

4.4.4. Token Request and Issuance

SAML tokens are requested by sending a WS-Trust Request Security Token (RST) message. In the RST message, the STS receives information about the type of token the client is requesting, the intended target for that token based on the service to which the client is sending its request message, and the credentials presented for authentication. In response to an RST, the STS issues a security token in a WS-Trust Request Security Token Response (RSTR) message. The RSTR contains the requested security token, and in this implementation, it also includes a proof token

containing a copy of a symmetric key to use when signing and encrypting messages for the service.

4.4.5. Overview of Key Exchange

Key exchange is the process by which two parties securely establish a secret key. The secret key can be used as proof-of-possession, or to encrypt and sign messages. If the client and service have no means to directly exchange keys with each other in a secure manner, then another party must facilitate the key exchange. In this implementation, the STS facilitates key exchange between the client and the service by creating a high entropy symmetric key for both parties.

Each party receives a copy of the symmetric key. The client, as the initiator of communication with a service, receives a copy of the symmetric key as plaintext, assuming that communication with the STS is secured. In this scenario, the copy of the symmetric key is included inside a proof token called a RequestedProofToken. The service receives a copy of the symmetric key as part of the security token requested by the client from the STS to communicate with that service. The service's copy of the symmetric key is encrypted with the public key from its X.509 certificate so only that service, as the holder of the private key that is paired with the public key, is the only one capable of decrypting its copy of the symmetric key.

Additionally, the STS signs the security token containing the symmetric key to establish data integrity and data origin authentication. This mitigates a man-in-the-middle attack when an attacker substitutes the symmetric key created by the STS with a key of their own, in an attempt to compromise secure communication between the client and the service. Also, it binds the identity of the STS to security tokens that it issues, so that the service has assurance that the token came from a trusted STS.

4.5. Design

There are 6 steps of the implementation:

1. The client requests an SAML token from the Identity STS.
2. The STS processes the RST and sends a response
3. The client requests an SAML token from the Credit Rating STS

4. The STS processes the RST and sends a response
5. The client sends a request to the bank service
6. The service processes the request and sends a response

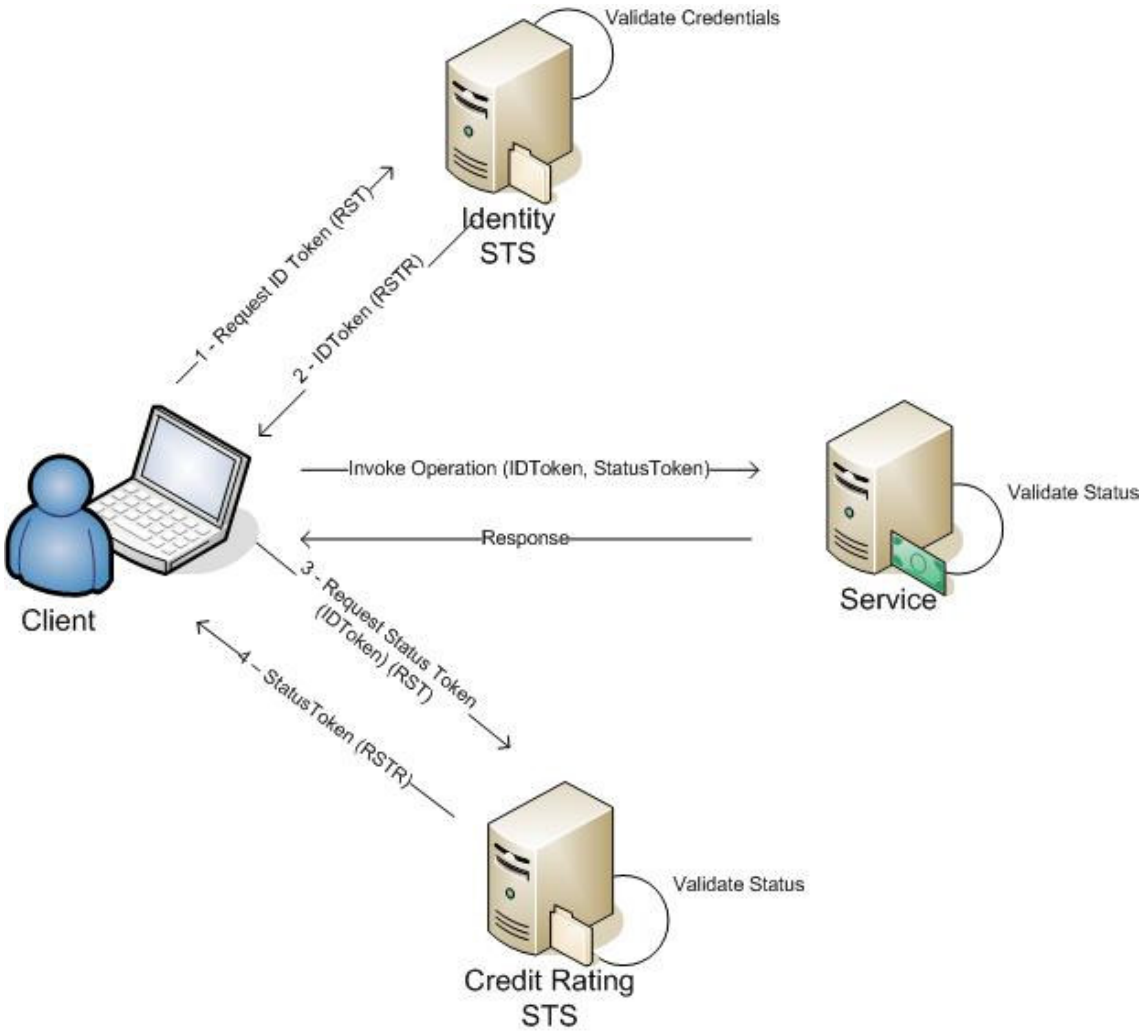

Figure 4.3 Scenario Flow

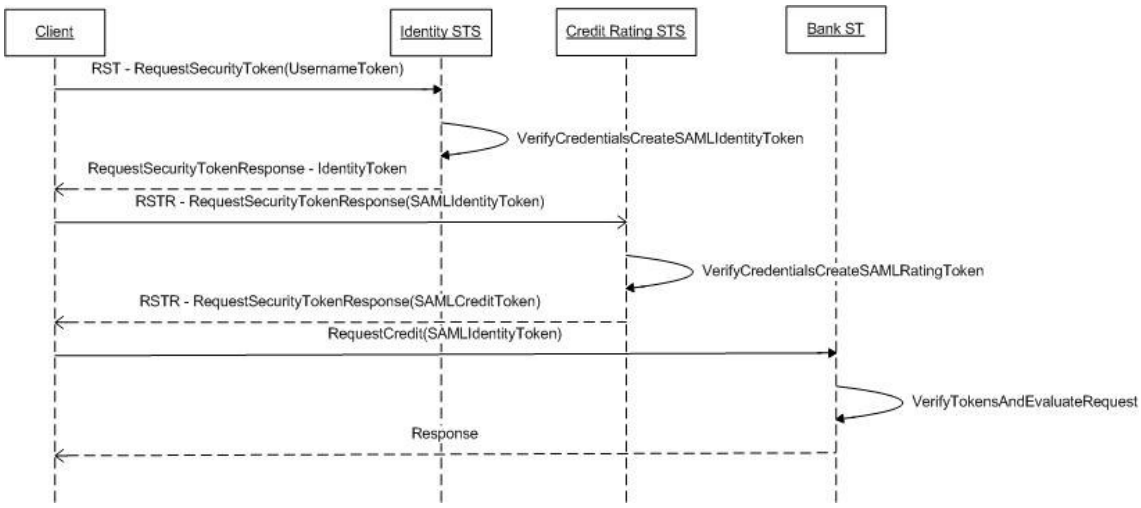

Figure 4.4 Sequence Diagram of the Scenario

4.5.1. Process Step 1 – The Client Requests an SAML Token from the Identity STS

In the first step, the client initializes a username token which would be sent in plain text. Using the username and password information supplied here, the STS would be able to authenticate the user. We used plain text, since some directory server implementations which could be used as data store for client information requires plaintext passwords for client authentication. We will be using message layer security before sending the message containing this sensitive information.,

As the second step, the client initializes an RST message to request a security token from the STS. The type of token requested, the target of the the token specified as an endpoint reference, and the lifetime of the requested token is included in the message. While the client may ask for a desired lifetime, it is up to the the STS to decide on the lifetime "(ANDERSON, 2005)".

```
<wst:RequestSecurityToken
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
        <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
                1.1#SAMLV1.1</wst:TokenType>
        <wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
        </wst:RequestType>
        <wst:Lifetime>
                <wsu:Expires>2006-12-08T21:54:46Z</wsu:Expires>
        </wst:Lifetime>
</wst:RequestSecurityToken>
```

Example 4.1 RST Message Example

<wst:TokenType> element in the RST message, identifies the type of token requested to be issued. In this example, we ask for a token in conformance with the SAML token profile. The <wst:Lifetime> element defines the requested lifetime for the token. The token can be cached depending on the archiecture of the application, and can be used over multiple requests for its lifetime.

In step three, the client includes a derived key token in the request message that is encrypted with a wrapped symmetric encryption key. The wrapped symmetric key is encrypted with the service's X.509 certificate public key. This key is referred to as an encrypted key. Accompanied by a valid UsernameToken, data origin authentication is provided when the client uses the derived key token to sign the message.

As the last step, the prepared message is sent to the STS.

4.5.2 The STS Processes the RST and Sends a Response

Upon receiving the message, the STS decrypts the message. First the symmetric key is decripted with the STS's private key. Next the symmetric key is used to decrypt the derived key that is used to encrypt and sign the rest of the message. The derived key token, that has been decripted, is used to verify message integrity. Also the information in the username token included in the message is validated against a data store such as a directory service.

Once the credentials are validated, an SAML token is initialized by the STS. The STS signs the token using its X.509 certificate public key to provide data integrity and to provide proof that the token was indeed issued by the STS.

```
<saml:Assertion AssertionID="SecurityToken-07dc7c16-9a42-4100-ad21-13013b975f3c"
        MajorVersion="1" MinorVersion="1"
        Issuer="http://localhost/SamlSecurityTokenService/SamlTokenIssuer.ashx"
        IssueInstant="2005-11-29T22:36:03Z"
        xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
        <saml:Conditions NotBefore="2005-11-29T22:36:03Z" NotOnOrAfter="2005-11-
                30T02:36:03Z">
                <saml:AudienceRestrictionCondition>
                        <saml:Audience>

        http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
                        </saml:Audience>
                </saml:AudienceRestrictionCondition>
        </saml:Conditions>
        <saml:AuthenticationStatement
                AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
                AuthenticationInstant="2005-11-29T22:36:03Z">
                <saml:Subject>
                        <saml:NameIdentifier
        Format="http://schemas.xmlsoap.org/ws/2004/01/Federation/username">RDAL
APTOP02\wseuser</saml:NameIdentifier>
                        <saml:SubjectConfirmation>
                <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-
ofkey</saml:ConfirmationMethod>
                        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                                <xenc:EncryptedKey

        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                                <xenc:EncryptionMethod
                                Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
1_5" />
                                <KeyInfo
xmlns="http://www.w3.org/2000/09/xmldsig#">
                                <wsse:SecurityTokenReference>
                                <wsse:KeyIdentifier ValueType="http://docs.oasisopen.
                        org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1"
                        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soapmessage-security-
1.0#Base64Binary">aAI1zTqHbhsUN6j2HsIefWcHODs=
                                </wsse:KeyIdentifier>
                                </wsse:SecurityTokenReference>
                        </KeyInfo>
                        <xenc:CipherData>
        <xenc:CipherValue>cSrRHLh8DWeELh5Naf34z515OIs0sw6hp4/zUIOnbOFNj1lxQ
dXCTi7z3aaLu4Xiws8vF3YdzD9LD/bQ1+QzzI7qcR4eDLNnxjZU87DkCBxI4ygqyB+Mx4J
2lKLYl+rxIOVOVcjbd64/YngQu5AgZKBxNZv7GIcla0d3Ikebyr4=</xenc:CipherValue>
```

Example 4.2 Sample SAML Token

```
                        </xenc:CipherData>
                                </xenc:EncryptedKey>
                        </ds:KeyInfo>
                        </saml:SubjectConfirmation>
                </saml:Subject>
                <saml:SubjectLocality IPAddress="192.168.0.10"
        DNSAddress="CLIENTHOST" />
        </saml:AuthenticationStatement>
        <saml:AttributeStatement>
                <saml:Subject>
                        <saml:NameIdentifier
        Format="http://schemas.xmlsoap.org/ws/2004/01/Federation/username">RDAL
APTOP02\wseuser
                        </saml:NameIdentifier>
                        <saml:SubjectConfirmation>
                <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-
ofkey
                </saml:ConfirmationMethod>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

        <xenc:EncryptedKeyxmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
        <xenc:EncryptionMethodAlgorithm="http://www.w3.org/2001/04/xmlenc#rsa-
1_5" />
                <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                        <wsse:SecurityTokenReference>
                                <wsse:KeyIdentifier ValueType="http://docs.oasisopen.
org/wss/oasis-wss-soap-message-security-1.1#ThumbprintSHA1"
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soapmessage-
security-1.0#Base64Binary">aAI1zTqHbhsUN6j2HsIefWcHODs=
                                </wsse:KeyIdentifier>
                        </wsse:SecurityTokenReference>
        </KeyInfo>
        <xenc:CipherData>
        <xenc:CipherValue>cSrRHLh8DWeELh5Naf34z515OIs0sw6hp4/zUIOnbOFNj
1lxQdXCTi7z3aaLu4Xiws8vF3YdzD9LD/bQ1+QzzI7qcR4eDLNnxjZU87DkCBxI4yg
qyB+Mx4J2lKLYl+rxIOVOVcjbd64/YngQu5AgZKBxNZv7GIcla0d3Ikebyr4=
        </xenc:CipherValue>
        </xenc:CipherData>
        </xenc:EncryptedKey>
        </ds:KeyInfo>
        </saml:SubjectConfirmation>
        </saml:Subject>
        <saml:Attribute AttributeName="group"
        AttributeNamespace="http://schemas.xmlsoap.org/ws/2004/01/Federation/group
">
```

Example 4.2 (Continued)

```
                <saml:AttributeValue>BUILTIN\Users
                </saml:AttributeValue>
        </saml:Attribute>
        </saml:AttributeStatement>
        <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
                <SignedInfo>
                        <ds:CanonicalizationMethod
                                Algorithm="http://www.w3.org/2001/10/xml-excc14n#"
                                        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
/>
                                <SignatureMethod
                                Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
                        <Reference URI="#SecurityToken-07dc7c16-9a42-4100-ad21-
13013b975f3c">
                        <Transforms>
                        <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#envelopedsignature"/>
                        <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#" />
                        </Transforms>
                <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
/>
                <DigestValue>k2PBIDmwJLlQIh/GA4bVPgk1544=</DigestValue>
        </Reference>
        </SignedInfo>
        <SignatureValue>CZPQJvc2vpjKyio6EEJTQShHGlJff1hZubOWSDCbRrbSFk
K53fOp9Ugulfel6vvs9yZfFJB2ieRAPK3ywUrRWTKvO7vjXP9HRGgrvqGC2PpQNA
EYn7ciBkLM+VoJV5vWfIopVevEYvnxFMFZlJTlLSOr0n+GWYaZuUYFJECfHaE=
</SignatureValue>
        <KeyInfo>
                <wsse:SecurityTokenReference>
                <wsse:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/oasiswss-
soap-message-security-1.1#ThumbprintSHA1" EncodingType="http://docs.oasisopen.
org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary">
                                vLfjdZyxqw+ztcYkVVQX8wyXp5o=
                </wsse:KeyIdentifier>
                </wsse:SecurityTokenReference>
        </KeyInfo>
        </Signature>
</saml:Assertion>
```

Example 4.2 (Continued)

The STS generates a high entropy symmetric key that can be used to provide proof-of-possession to confirm the subject's identity and provides a key for the client and serviceto sign and encrypt messages. The symmetric key is a hash computation, created by the STS using a PSHA1 algorithm. The STS uses an entropy value it generates, plus an entropy value provided by the client in the RST message as values for key generation.

The STS generates the key and encrypts a copy of it with the target service's X.509 certificate public key. The STS determines which X.509 certificate it uses to encrypt the symmetric key from a mapping that it maintains in its configuration file. This mapping provides traceability from the scope of the security token requested by the client in the RST message to the SHA1 Thumbprint of the X.509 certificate used for encrypting token data. The STS includes the encrypted symmetric key as a claim in the SAML token, and includes a copy of the symmetirc key in a proof token returned back to the client in the RSTR.

The STS includes the SAML token and a proof token in a RSTR message back to the client. The proof token contains a copy of the symmetric key that was added to the SAML token. Unlike the copy of the symmetric key in the SAML token, the copy in the proof token is not encrypted using the target services X.509 certificate public key.Instead, the key in the proof token is plaintext, though the RSTR is secured. The client caches the proof token to derive encryption and signing keys for securing messages exchanged with the service.

4.5.3. The client requests an SAML token from the Credit Rating STS

The SAML token received from the Identity STS is attached to the RST message which is going to sent to the Credit Rating STS. The client's copy of the symmetric key contained in the proof token received with the message is used to derive a signing key to sign the message. The signing key is created using PSHA1 algorithm which produces a hash value from the symmetric key, and a concatenation of a label and a nonce. Key derivation allows fresh key material to be used each time messages are signed to reduce the success of offline analysis by attackers. Another key is derived from the symmetric key. The encryption key is used to encrypt sensitive parts of the message such as message body and signature. The prepared message is sent to the Credit Rating STS.

4.5.4. The Credit Rating STS Processes the RST and Sends a Response

The message is decrypted as explained in first paragraph of section 4.5.2. The service computes the derived signing key from the symmetric key in the SAML token and verifies the message signature. This provides data origin authentication, as it gives the service assurance that the message was not tampered with during transit, and that it originated from the client as the holder of the symmetric key included in the SAML token.

Next the Credit Rating STS checks the certificate identifier of the X.509 certificate the Identity STS used to sign the SAML token against a list of trusted token issuers. The Credit Rating STS also verifies that the required parts of the token are present and the validity period of the token. The service prepares a new extended SAML token which also includes the credit rating of the user.

A new RSTR message is prepared as explained in section 4.5.2. This new message includes the SAML token prepared by the Credit Rating STS. The message is sent back to the client.

4.5.5. The client sends a request to the bank service

Upon receiving the RSTS from the Credit Rating STS, the client receives the second SAML token with the custom assertion of the user's credit rating. The client prepares a service request from the bank web service. For preparing this message, the process explained in section 4.5.3 is repeated only this time the symmetric key prepared using the information in the second SAML token prepared by the Credit Rating STS is used. The second SAML token is attached to this message and sent to the bank web service.

4.5.6. The Service Processes the Request and Sends a Response

The service computes the derived key used to encrypt the message from the symmetric key in the SAML token. The derived encryption key is then used to decrypt the request message signature and body. Next, the service computes the derived signing

key from the symmetric key in the SAML token and verifies the message signature. This provides data origin authentication, as it gives the service assurance that the message was not tampered with during transit, and that it originated from the client as the holder of the symmetric key included in the SAML token.

The bank web service checks the certificate identifier of the X.509 certificate the STS used to sign the SAML token against a list of trusted token issuers. If the SAML token is not signed by a trusted token issuer, an exception is thrown.

As the last step of the process, the service applies business logic using input from the SAML token, namely the identity credentials and the credit rating from the Credit Rating Service to decide on the outcome of the request. The response, signed and encrypted with keys derived from the symmetric key that the service obtained from the SAML token is sent back to the client.

CHAPTER 5

CONCLUSION

With this thesis, we have laid the foundations for a more comprehensive framework for securing Web services. Our proposed architecture demonstrates an example of trust brokering for various purposes. The fact that each of the actors are located in different security domains, can be considered a hint for implementing a federation framework for service oriented architectures. Also Web services single sign on can be implemented by caching the received SAML token.

There are other extension points which are outside the scope of this thesis. One such extension is implementing secure conversation between actors. Such a need would arise if the conversation between actors includes more than one interaction. Secure conversation can be implemented by establishing security context tokens and using these tokens for securing the conversation.

Throughout this thesis, we have assumed that the parties involved in the architecture, knew beforehand the requirements of other actors. This might not be the case when a large scale service oriented architecture is built. Policy frameworks such as Web Services Policy technology can be implemented for discovery of requirements of actors.

Other extensions such as support for different security tokens like Kerberos tokens can be added to this architecture for supporting other systems using them can be also added to this architecture.

REFERENCES

Anderson, S. et al, 2005, Web Services Trust Language (WS-Trust)

Atkinson, B. et al, 2002. “Web Services Security (WS-Security)”, p.2

IBM Corporation, Microsoft Corporation, 2002. Security in a Web Services World: A Proposed Architecture and Roadmap

OASIS - Organization for the Advancement of Structured Information Standards, 2006. Web Services Security SAML Token Profile 1.1, p.7

OASIS - Organization for the Advancement of Structured Information Standards, 2006. Web Services Security Username Token Profile 1.1, p.7

WEB_1, 2006. Web Services Glossary, 06/07/2006, http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

WEB_2, 2006. SoapRPC.com: Glossary, 06/07/2006, http://www.soaprpc.com/glossary.html

WEB_3, 2006. webservices.xml.com: WS-Trust: Interoperable Security for Web Services, 06/07/2006, http://webservices.xml.com/pub/a/ws/2003/06/24/ws-trust.html?page=1

APPENDIX A

GLOSSARY "(WEB_1 2006)"

Access

To interact with a system entity in order to manipulate, use, gain knowledge of, and/or obtain a representation of some or all of a system entity's resources. [RFC 2828]
access control

Protection of resources against unauthorized access; a process by which use of resources is regulated according to a security policy and is permitted by only authorized system entities according to that policy. [RFC 2828]

Access Control Information

Any information used for access control purposes, including contextual information. [X.812]

Contextual information might include source IP address, encryption strength, the type of operation being requested, time of day, etc. Portions of access control information may be specific to the request itself, some may be associated with the connection via which the request is transmitted, and others (for example, time of day) may be "environmental". [RFC 2829]

Access Rights

A description of the type of authorized interactions a subject can have with a resource. Examples include read, write, execute, add, modify, and delete. [WSIA Glossary]

Actor

A person or organization that may be the owner of agents that either seek to use Web services or provide Web services.

A physical or conceptual entity that can perform actions. Examples: people; companies; machines; running software. An actor can take on (or implement) one or more roles. An actor at one level of abstraction may be viewed as a role at a lower level of abstraction.

Agent

An agent is a program acting on behalf of a person or organization. (This definition is a specialization of the definition in [Web Arch]. It corresponds to the notion of software agent in [Web Arch].)

Anonymity

The quality or state of being anonymous, which is the condition of having a name or identity that is unknown or concealed. [RFC 2828]

Architecture

The software architecture of a program or computing system is the structure or structures of the system. This structure includes software components, the externally visible properties of those components, the relationships among them and the constraints on their use. (based on the definition of architecture in [Soft Arch Pract])

A software architecture is an abstraction of the run-time elements of a software system during some phase of its operation. A system may be composed of many levels of abstraction and many phases of operation, each with its own software architecture. [Fielding]

Artifact

A piece of digital information. An artifact may be any size, and may be composed of other artifacts. Examples of artifacts: a message; a URI; an XML document; a PNG image; a bit stream.

Asynchronous

An interaction is said to be asynchronous when the associated messages are chronologically and procedurally decoupled. For example, in a request-response interaction, the client agent can process the response at some indeterminate point in the future when its existence is discovered. Mechanisms to do this include polling, notification by receipt of another message, etc.

Attribute

A distinct characteristic of an object. An object's attributes are said to describe the object. Objects' attributes are often specified in terms of their physical traits, such as size, shape, weight, and color, etc., for real-world objects. Objects in cyberspace might have attributes describing size, type of encoding, network address, etc. [WSIA Glossary]

Audit Guard

An audit guard is a mechanism used on behalf of an owner that monitors actions and agents to verify the satisfaction of obligations.

Authentication

Authentication is the process of verifying that a potential partner in a conversation is capable of representing a person or organization.

Authorization

The process of determining, by evaluating applicable access control information, whether a subject is allowed to have the specified types of access to a particular resource. Usually, authorization is in the context of authentication. Once a subject is authenticated, it may be authorized to perform different types of access. [STG]

Binding

An association between an interface, a concrete protocol and a data format. A binding specifies the protocol and data format to be used in transmitting messages defined by the associated interface. [WSD Reqs]

The mapping of an interface and its associated operations to a particular concrete message format and transmission protocol.

See also SOAP binding.

Capability

A capability is a named piece of functionality (or feature) that is declared as supported or requested by an agent.

Choreography

A choreography defines the sequence and conditions under which multiple cooperating independent agents exchange messages in order to perform a task to achieve a goal state.

Web Services Choreography concerns the interactions of services with their users. Any user of a Web service, automated or otherwise, is a client of that service. These users may, in turn, be other Web Services, applications or human beings. Transactions among Web Services and their clients must clearly be well defined at the time of their execution, and may consist of multiple separate interactions whose composition constitutes a complete transaction. This composition, its message protocols, interfaces, sequencing, and associated logic, is considered to be a choreography. [WSC Reqs]

Component

A component is a software object, meant to interact with other components, encapsulating certain functionality or a set of functionalities. A component has a clearly defined interface and conforms to a prescribed behavior common to all components within an architecture. [CCA T&D]

A component is an abstract unit of software instructions and internal state that provides a transformation of data via its interface. [Fielding]

A component is a unit of architecture with defined boundaries.

Confidentiality

Assuring information will be kept secret, with access limited to appropriate persons. [NSA Glossary]

Configuration

A collection of properties which may be changed. A property may influence the behavior of an entity.

Connection

A transport layer virtual circuit established between two programs for the purpose of communication. [RFC 2616]

Control

To cause a desired change in state. Management systems may control the life cycle of manageable Web services or information flow such as messages.

Conversation

A Web service conversation involves maintaining some state during an interaction that involves multiple messages and/or participants.

Credentials

Data that is transferred to establish a claimed principal identity. [X.800]

Delivery Policy

A delivery policy is a policy that constrains the methods by which messages are delivered by the message transport.

Digital Signature

A value computed with a cryptographic algorithm and appended to a data object in such a way that any recipient of the data can use the signature to verify the data's origin and integrity. (See: data origin authentication service, data integrity service, digitized signature, electronic signature, signer.) [RFC 2828]

Discovery

The act of locating a machine-processable description of a Web service-related resource that may have been previously unknown and that meets certain functional criteria. It involves matching a set of functional and other criteria with a set of resource descriptions. The goal is to find an appropriate Web service-related resource.

Discovery Service

A discovery service is a service that enables agents to retrieve Web services-related resource description.

Document

Any data that can be represented in a digital form. [UeB Glossary]

Electronic Data Interchange (EDI)

The automated exchange of any predefined and structured data for business among information systems of two or more organizations. [ISO/IEC 14662]

Domain

A domain is an identified set of agents and/or resources that is subject to the constraints of one of more policies.

Encryption

Cryptographic transformation of data (called "plaintext") into a form (called "ciphertext") that conceals the data's original meaning to prevent it from being known or used. If the transformation is reversible, the corresponding reversal process is called "decryption", which is a transformation that restores encrypted data to its original state. [RFC 2828]

End Point

An association between a binding and a network address, specified by a URI, that may be used to communicate with an instance of a service. An end point indicates a specific location for accessing a service using a specific protocol and data format. [WSD Reqs]

Gateway

An agent that terminates a message on an inbound interface with the intent of presenting it through an outbound interface as a new message. Unlike a proxy, a gateway receives messages as if it were the final receiver for the message. Due to possible mismatches between the inbound and outbound interfaces, a message may be modified and may have some or all of its meaning lost during the conversion process. For example, an HTTP PUT has no equivalent in SMTP.

Note: a gateway may or may not be a SOAP node; however a gateway is never a SOAP intermediary, since gateways terminate messages and SOAP intermediaries relay them instead. Being a gateway is typically a permanent role, whilst being a SOAP intermediary is message specific.

Idempotent

Property of an interaction whose results and side-effects are the same whether it is done one or multiple times. [RFC 2616]

Safe interactions are inherently idempotent.

Identifier

An identifier is an unambiguous name for a resource.

Initial SOAP Sender

The SOAP sender that originates a SOAP message at the starting point of a SOAP message path.

Integrity

Assuring information will not be accidentally or maliciously altered or destroyed. [NSA Glossary]

Loose Coupling

Coupling is the dependency between interacting systems. This dependency can be decomposed into real dependency and artificial dependency:

Real dependency is the set of features or services that a system consumes from other systems. The real dependency always exists and cannot be reduced.

Artificial dependency is the set of factors that a system has to comply with in order to consume the features or services provided by other systems. Typical artificial dependency factors are language dependency, platform dependency, API dependency, etc. Artificial dependency always exists, but it or its cost can be reduced.

Loose coupling describes the configuration in which artificial dependency has been reduced to the minimum.

Manageable Service

A Web service becomes a manageable service with additional semantics, policy statements, and monitoring and control (or management) capabilities (exposed via a management interface) all for the purpose of managing the service.

Management

The utilization of the management capabilities by the management system in order to perform monitoring of values, tracking of states and control of entities in order to produce and maintain a stable operational environment.

Management Capability

Capabilities that a Web service has for the purposes of controlling or monitoring the service, and that can be exposed to a management system for the sole purpose of managing the service.

Management Interface

Interface through which the management capabilities of a service are exposed.

Management Policy

Policy associated with a Web service solely for the purpose of describing the management obligations and permissions for the service.

Management Semantics

The management semantics of a service augment the semantics of a service with management-specific semantics. These management semantics form the contract between the provider entity and the requester entity that expresses the effects and requirements pertaining to the management and management policies for a service.

Message

A message is the basic unit of data sent from one Web services agent to another in the context of Web services.

The basic unit of communication between a Web service and a requester: data to be communicated to or from a Web service as a single logical transmission. [WSD Reqs]

See also SOAP message.

Message Correlation

Message correlation is the association of a message with a context. Message correlation ensures that the requester agent can match the reply with the request, especially when multiple replies may be possible.

Message Exchange Pattern (MEP)

A Message Exchanage Pattern (MEP) is a template, devoid of application semantics, that describes a generic pattern for the exchange of messages between agents. It describes the relationships (e.g., temporal, causal, sequential, etc.) of multiple messages exchanged in conformance with the pattern, as well as the normal and abnormal termination of any message exchange conforming to the pattern.

See SOAP message exchange pattern (MEP).

Message Receiver

A message receiver is an agent that receives a message.

Message Reliability

Message reliability is the degree of certainty that a message will be delivered and that sender and receiver will both have the same understanding of the delivery status.

Message Sender

A message sender is the agent that transmits a message.

Message Transport

A message transport is a mechanism that may be used by agents to deliver messages.

Non-Repudiation

Method by which the sender of data is provided with proof of delivery and the recipient is assured of the sender's identity, so that neither can later deny having processed the data. [INFOSEC Glossary]

Obligation

An obligation is a kind of policy that prescribes actions and/or states of an agent and/or resource.

Operation

A set of messages related to a single Web service action. [WSD Reqs]

Orchestration

An orchestration defines the sequence and conditions in which one Web service invokes other Web services in order to realize some useful function. I.e., an orchestration is the pattern of interactions that a Web service agent must follow in order to achieve its goal.

Permission

A permission is a kind of policy that prescribes the allowed actions and states of an agent and/or resource.

Permission Guard

A permission guard is a mechanism deployed on behalf of an owner to enforce permission policies.

Person Or Organization

A person or organization may be the owner of agents that provide or request Web services.

Policy

A policy is a constraint on the behavior of agents or person or organization.

Policy Guard

A policy guard is a mechanism that enforces one or more policies. It is deployed on behalf of an owner.

Principal

A system entity whose identity can be authenticated. [X.811]

Privacy Policy

A set of rules and practices that specify or regulate how a person or organization collects, processes (uses) and discloses another party's personal data as a result of an interaction.

Provider Agent

An agent that is capable of and empowered to perform the actions associated with a service on behalf of its owner — the provider entity.

Provider Entity

The person or organization that is providing a Web service.

Protocol

A set of formal rules describing how to transmit data, especially across a network. Low level protocols define the electrical and physical standards to be observed, bit- and byte-ordering and the transmission and error detection and correction of the bit stream. High level protocols deal with the data formatting, including the syntax of messages, the terminal to computer dialogue, character sets, sequencing of messages etc. [FOLDOC]

Proxy

An agent that relays a message between a requester agent and a provider agent, appearing to the Web service to be the requester.

Quality Of Service

Quality of Service is an obligation accepted and advertised by a provider entity to service consumers.

Reference Architecture

A reference architecture is the generalized architecture of several end systems that share one or more common domains. The reference architecture defines the infrastructure common to the end systems and the interfaces of components that will be included in the end systems. The reference architecture is then instantiated to create a

software architecture of a specific system. The definition of the reference architecture facilitates deriving and extending new software architectures for classes of systems. A reference architecture, therefore, plays a dual role with regard to specific target software architectures. First, it generalizes and extracts common functions and configurations. Second, it provides a base for instantiating target systems that use that common base more reliably and cost effectively. [Ref Arch]

Registry

Authoritative, centrally controlled store of information.

Requester Agent

A software agent that wishes to interact with a provider agent in order to request that a task be performed on behalf of its owner — the requester entity.

Requester Entity

The person or organization that wishes to use a provider entity's Web service.

Safe

Property of an interaction which does not have any significance of taking an action other than retrieval of information. [RFC 2616]

security administration

Configuring, securing and/or deploying of systems or applications enabling a security domain.

Security Architecture

A plan and set of principles for an administrative domain and its security domains that describe the security services that a system is required to provide to meet the needs of its users, the system elements required to implement the services, and the performance levels required in the elements to deal with the threat environment. A complete security architecture for a system addresses administrative security, communication security, computer security, emanations security, personnel security, and physical security, and prescribes security policies for each. A complete security architecture needs to deal with both intentional, intelligent threats and accidental threats.

A security architecture should explicitly evolve over time as an integral part of its administrative domain's evolution. [RFC 2828]

Security Auditing

A service that reliably and securely records security-related events producing an audit trail enabling the reconstruction and examination of a sequence of events. Security events could include authentication events, policy enforcement decisions, and others. The resulting audit trail may be used to detect attacks, confirm compliance with policy, deter abuse, or other purposes.

Security Domain

An environment or context that is defined by security models and a security architecture, including a set of resources and set of system entities that are authorized to access the resources. One or more security domains may reside in a single administrative domain. The traits defining a given security domain typically evolve over time. [RFC 2828]

Security Mechanism

A process (or a device incorporating such a process) that can be used in a system to implement a security service that is provided by or within the system.

Security Model

A schematic description of a set of entities and relationships by which a specified set of security services are provided by or within a system. [RFC 2828]

Security Policy

A set of rules and practices that specify or regulate how a system or organization provides security services to protect resources. Security policies are components of security architectures. Significant portions of security policies are implemented via security services, using security policy expressions. [RFC 2828]

Security Policy Expression

A mapping of principal identities and/or attributes thereof with allowable actions. Security policy expressions are often essentially access control lists. [STG]

Security Service

A processing or communication service that is provided by a system to give a specific kind of protection to resources, where said resources may reside with said system or reside with other systems, for example, an authentication service or a PKI-based document attribution and authentication service. A security service is a superset of AAA services. Security services typically implement portions of security policies and are implemented via security mechanisms. [RFC 2828]

Service

A service is an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requester entities. To be used, a service must be realized by a concrete provider agent.
WSDL service: A collection of end points. [WSD Reqs]

See Web service.

Service Description

A service description is a set of documents that describe the interface to and semantics of a service.

Service Interface

A service interface is the abstract boundary that a service exposes. It defines the types of messages and the message exchange patterns that are involved in interacting with the service, together with any conditions implied by those messages.

A logical grouping of operations. An interface represents an abstract service type, independent of transmission protocol and data format. [WSD Reqs]

Service Intermediary

A service intermediary is a Web service whose main role is to transform messages in a value-added way. (From a messaging point of view, an intermediary processes messages en route from one agent to another.) Specifically, we say that a service intermediary is a service whose outgoing messages are equivalent to its incoming messages in some application-defined sense.

See SOAP intermediary.

Service Provider

See provider agent and provider entity. See also the discussion about service provider in [WS Arch].

Service Requester

See requester agent and requester entity. See also the discussion about service requester in [WS Arch].

Service Role

An abstract set of tasks which is identified to be relevant by a person or organization offering a service. Service roles are also associated with particular aspects of messages exchanged with a service.

Service Semantics

The semantics of a service is the behavior expected when interacting with the service. The semantics expresses a contract (not necessarily a legal contract) between the provider entity and the requester entity. It expresses the effect of invoking the service. A service semantics may be formally described in a machine readable form, identified but not formally defined, or informally defined via an out of band agreement between the provider and the requester entity.

Service-Oriented Architecture

A set of components which can be invoked, and whose interface descriptions can be published and discovered.

Session

A lasting interaction between system entities, often involving a user, typified by the maintenance of some state of the interaction for the duration of the interaction. [WSIA Glossary]

Such an interaction may not be limited to a single connection between the system entities.

SOAP

The formal set of conventions governing the format and processing rules of a SOAP message. These conventions include the interactions among SOAP nodes generating and accepting SOAP messages for the purpose of exchanging information along a SOAP message path.

SOAP Application

A software entity that produces, consumes or otherwise acts upon SOAP messages in a manner conforming to the SOAP processing model.

SOAP Binding

The formal set of rules for carrying a SOAP message within or on top of another protocol (underlying protocol) for the purpose of exchange. Examples of SOAP bindings include carrying a SOAP message within an HTTP entity-body, or over a TCP stream.

SOAP Body

A collection of zero or more element information items targeted at an ultimate SOAP receiver in the SOAP message path.

SOAP Envelope

The outermost element information item of a SOAP message.

SOAP Fault

A SOAP element information item which contains fault information generated by a SOAP node.

SOAP Feature

An extension of the SOAP messaging framework typically associated with the exchange of messages between communicating SOAP nodes. Examples of features include "reliability", "security", "correlation", "routing", and the concept of message exchange patterns.

SOAP Header

A collection of zero or more SOAP header blocks each of which might be targeted at any SOAP receiver within the SOAP message path.

SOAP Header Block

An element information item used to delimit data that logically constitutes a single computational unit within the SOAP header. The type of a SOAP header block is identified by the fully qualified name of the header block element information item.

SOAP Intermediary

A SOAP intermediary is both a SOAP receiver and a SOAP sender and is targetable from within a SOAP message. It processes the SOAP header blocks targeted at it and acts to forward a SOAP message towards an ultimate SOAP receiver.

SOAP Message

The basic unit of communication between SOAP nodes.

SOAP Message Exchange Pattern (MEP)

A template for the exchange of SOAP messages between SOAP nodes enabled by one or more underlying SOAP protocol bindings. A SOAP MEP is an example of a SOAP feature.

SOAP Message Path

The set of SOAP nodes through which a single SOAP message passes. This includes the initial SOAP sender, zero or more SOAP intermediaries, and an ultimate SOAP receiver.

SOAP Node

The embodiment of the processing logic necessary to transmit, receive, process and/or relay a SOAP message, according to the set of conventions defined by this recommendation. A SOAP node is responsible for enforcing the rules that govern the exchange of SOAP messages. It accesses the services provided by the underlying protocols through one or more SOAP bindings.

SOAP Receiver

A SOAP node that accepts a SOAP message.

SOAP Role

A SOAP node's expected function in processing a message. A SOAP node can act in multiple roles.

SOAP Sender

A SOAP node that transmits a SOAP message.

State

A set of attributes representing the properties of a component at some point in time.

Synchronous

An interaction is said to be synchronous when the participating agents must be available to receive and process the associated messages from the time the interaction is initiated until all messages are actually received or some failure condition is determined. The exact meaning of "available to receive the message" depends on the characteristics of the participating agents (including the transfer protocol it uses); it may, but does not necessarily, imply tight time synchronization, blocking a thread, etc.

System Entity

An active element of a computer/network system. For example, an automated process or set of processes, a subsystem, a person or group of persons that incorporates a distinct set of functionality. [RFC 2828]

Transaction

Transaction is a feature of the architecture that supports the coordination of results or operations on state in a multi-step interaction. The fundamental characteristic of a transaction is the ability to join multiple actions into the same unit of work, such that the actions either succeed or fail as a unit.

Ultimate SOAP **Receiver**

The SOAP receiver that is a final destination of a SOAP message. It is responsible for processing the contents of the SOAP body and any SOAP header blocks targeted at it. In some circumstances, a SOAP message might not reach an ultimate SOAP receiver, for example because of a problem at a SOAP intermediary. An ultimate SOAP receiver cannot also be a SOAP intermediary for the same SOAP message.

Usage Auditing

Service that reliably and securely records usage-related events producing an audit trail enabling the reconstruction and examination of a sequence of events. Usage events could include resource allocation events and resource freeing events.

Web Service

There are many things that might be called "Web services" in the world at large. However, for the purpose of this Working Group and this architecture, and without prejudice toward other definitions, we will use the following definition:

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.