WEB SERVICES BASED SECURITY APPLICATION FRAMEWORK MODEL

by

Ertan DENİZ

August 2009

WEB SERVICES BASED SECURITY APPLICATION FRAMEWORK MODEL

by

Ertan DENİZ

A thesis submitted to

the Graduate Institute of Sciences and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

August 2009
Istanbul, Turkey

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____
Assist. Prof. Tuğrul YANIK
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____
Assist. Prof. Tuğrul YANIK
Supervisor

Examining Committee Members

Assist. Prof. Tuğrul YANIK _____

Assist. Prof. Mehmet ŞEVKLİ _____

Assist. Prof. Veli HAKKOYMAZ _____

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

_____
Assoc. Prof. Nurullah ARSLAN
Director

WEB SERVICES BASED SECURITY APPLICATION FRAMEWORK (WSSAF) MODEL

Ertan DENİZ

M. S. Thesis - Computer Engineering
August 2009

Supervisor: Assist. Prof. Tuğrul YANIK

ABSTRACT

Today we have a huge and globally distributed network called the Internet. All business domains like e-government agencies, universities, hospitals store and process their information resources internally, and desire to publish some of these resources as services giving restricted access to the others over the Internet. Web services based applications establish an important part of the next generation distributed computing applications. Using web services is the most dominant way for enterprises to interoperate and adopt Service Oriented Architecture (SOA). SOA applications provide services either to the end user applications or to other services in intranet or in the internet. Security in SOA is a deciding factor for many enterprises when moving to SOA. Within the past years, important steps have been taken in the development of Web Services Security standards and applying these to real life projects. Web Services Security Standards have been widely accepted with new and strong features that help to satisfy the security requirements in SOA applications.

This thesis aims to provide a security application framework which all the security features provided by secure web services to secure web services in the framework's container. The framework utilizes the power of the web services which can be summarized

as standards based application integration, being loosely coupled and platform and language independent development. Our framework has a relational repository model that stores security configurations of the web services in the container and other security related information. The repository model can be accessed by a central domain specific Data service. We are presenting the principles and the architectural model of the framework. The proposed framework models all the security related functions like Authentication, Authorization, Auditing, Exception Management and Reporting as web services and adds a layer on top of Web Services development toolkit called Microsoft Windows Communication Foundation which provides standards based functionality. Web Service configurations are arranged by using a Web Management Console. WSSAF utilizes the power of web services, secures web services using other secure web services, is lightweight and distributable, is easy configurable and cost effective and helps in adopting the SOA vision. We expect that WSSAF will encourage the business domains to make an easy transition to the Web Services world. Finally, a case study has been presented to test and show the feasibility of the framework.

Keywords : SOA (Service Oriented Architecture), Web Services, Web Services Security, Security Framework

WEB SERVİS TEMELLİ GÜVENLİK ALT YAPI TEKNOLOJİSİ MODELİ

Ertan DENİZ

Yüksek Lisan Tezi – Bilgisayar Mühendisliği
Temmuz 2009

Tez Yöneticisi: Yrd. Doç. Dr. Tuğrul YANIK

ÖZ

Günümüzde Internet olarak adlandırılan, dağıtık yapıda ve büyük bir bilgi ve iletişim ağına sahibiz. Devlete bağlı organizasyonlar, üniversite ve hastane gibi kurumlar, bilgi kaynaklarını kendi imkanları ile saklar ve üzerinde işlem yaparlar. Bazı bilgileri de, yetki dahilinde diğer kullanıcılara birtakım servisler aracılığı ile kullanıma açmak istemektedirler. Web servis uygulamaları, yeni nesil dağıtık uygulamalarının önemli bir bölümünü oluşturmaktadır. Web servislerinin kullanımı, diğer kurumlar ile entegrasyon sürecinde ve servis temelli mimariye geçişte de en çok tercih edilen yöntemdir. Servis temelli mimari ile geliştirilen uygulamalar, Internet ve Intranet ortamında, hem son kullanıcılara hem de diğer servislere hizmet sunarlar. Servis temelli mimari de güvenlik, bu mimariye geçişte, büyük öneme sahip ve karar verici etkisi bulunmaktadır. Geçen yıllar içinde, Web servis güvenliği ile ilgili standartların geliştirilmesi ve gerçek hayat projelerinde kullanılması yönünde büyük adımlar atıldı. Web servis güvenliği standartları, Servis temelli uygulamaların güvenlik gereksinimlerini karşılamasına destek olacak yeni ve güçlü özellikleri ile, daha geniş kabul görmeye başlamıştır.

Tezimizde, güvenlik fonksiyonlarının web servisleri olarak sunulduğu ve tanımlı olan web servislerini güvenli hale getirmeyi hedefleyen bir alt yapı teknoloji modeli

sağlamak hedeflenmiştir. Alt yapı teknolojisi, web servislerinin gücünden yararlanmaktadır. Web servislerinin güçlü olduğu özellikler; standartlar üzerine kurulu uygulama entegrasyonu, birbirlerine bağımlılıkları düşük olması, işletim platformu ve geliştirme dili açısından bağımsızlık şeklinde özetlenebilir. Alt yapı teknolojisi, tanımlı web servislerinin güvenlik konfigürasyonlarının ve güvenlik ile ilgili bilgilerin saklandığı bir ilişkisel veri saklama modeline sahiptir. Bu modele veri servisi ile erişilmektedir. Bu çalışmamızda, alt yapı teknolojisinin prensipleri ve mimari modeli sunulmaktadır. Önerilen alt yapı teknoloji modeli ile, Kimlik doğrulama, yetki belirleme, loglama, istisna-hata yönetimi ve raporlama gibi güvenlik fonksiyonları web servisleri olarak modellenir. Web Servis geliştirme ile ilgili standart fonksiyonların sunulduğu, Microsoft Windows Communication Foundation (WCF) alt yapısı üzerine bir katman olarak eklenmiştir. Web servis konfigürasyonları, web yönetim aracı ile verilir. Alt yapı teknolojisi, web servislerinin gücünden istifade eder, web servislerini diğer güvenli web servislerini kullanarak güvenli hale getirir, hafif ve dağıtılabilir yapıdadır, kolay ayarlanabilir ve düşük maliyetlidir, Servis temelli mimari vizyonuna geçiş sürecinde katkı sağlar. Bu çalışmanın, kurumların Web servisleri dünyasına kolay geçiş yapma konusunda cesaret vermesini bekliyoruz. Son olarak, alt yapı teknolojisinin test edilmesi ve etkinliğinin gösterilmesi için bir test projesi sunulmuştur.

.

ACKNOWLEDGEMENT

I express sincere appreciation to Assist.Prof. Tuğrul YANIK for his guidance and insight throughout the research.

I express my thanks and appreciation to my family for their understanding, motivation and patience. Lastly, but in no sense the least, I am thankful to all colleagues and friends who made my stay at the university a memorable and valuable experience.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOL/ABBREVIATION	
CA	Certificate Authority
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
OASIS	Organization for the Advancement of Structured Information Standards
PKI	Public Key Infrastructure
RBAC	Role Based Access Control
SAML	Security Assertion Markup Language
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
UDDI	Universal Description, Discovery, and Integration
W3C	World Wide Web Consortium
WCF	Windows Communication Foundation
WS	Web Service
WS-Security	Web Services Security
WS-I	Web Services Interoperability Organization
WSDL	Web Services Description Language
XACML	eXtensible Access Control Markup Language

XKMS	XML Key Management Specification
XML	eXtensible Markup Language

CHAPTER 1

INTRODUCTION

Today we have a huge and globally distributed network called the Internet. All business domains like e-government agencies, universities, hospitals store and process their information resources internally, and desire to publish some of these resources as services giving restricted access to the others over the Internet.

To avoid the duplication of information one should prefer to access the information from its original resource. Accessing information resources anytime, anywhere, through various devices in a secure manner is one of the main goals of the E-Business world. Security is a non-functional requirement (Papazoglou, 2003) that affects the quality of services in any information system. On the other hand the users of information systems have rising expectations such as completing their business processes even though the needed information is in another domain. These requirements force the further development of secure information access and information exchange standards rapidly. The improvement of the standards causes the information systems to be re-architectured to reach the goal of being further interoperable.

Web services (Booth et al, 2004) are interoperable distributed software systems which publish standard interfaces and support message based access to help efficient service integration. Main technologies behind web services are XML (Bray 2008), SOAP (Gudgin et al, 2007), UDDI (Clement et al, 2004) and WSDL (Chinnici et al, 2007). XML and SOAP make it possible for web services to communicate with each other and their clients over HTTP (W3C Specification, 2009) protocol while UDDI makes it possible to

search for other web services and WSDL makes it possible to get a description on how to use the published web service.

Web services are the next generation of distributed computing applications and becoming an important way for enterprises interoperate. Web services are loosely coupled and platform independent services developed with language independent manner.

Service Oriented Architecture (SOA) (Papazoglou et al, 2007) is an architectural style to reuse and integrate existing systems for designing new generation applications. SOA applications provide services either to end user applications or other services distributed in the internet or intranet. The importance of Service Oriented Architecture (SOA) is increasing. Web services are the most dominant way for adopting SOA. (Sprott D. et al, 2003) SOA forces a migration from the current development approach of object oriented and Application Programming Interface (API) based applications to a more interoperable and loosely coupled Service based applications.

Web Services are simple and easy to implement. By using XML, reduction in cost and development time is achieved, but at the same time they dramatically increasing the security risks. Web service requests and responses are transmitted in plain text over a protocol such as HTTP. Using plain text transmission brings security risks. On the other hand, Security in the SOA applications has a great importance and a deciding factor for every organization when moving to SOA. In SOA, the non-functional requirements (Security, Performance, Availability, etc) of services should be described separately from their functional aspects (Business logic, Business Processes).

WS-Security (IBM and Microsoft, 2002) was developed by IBM, Microsoft, and VeriSign. These companies have submitted WS-Security specification to OASIS (OASIS Community), the international standards organization for e-business. OASIS has approved WS-Security as an OASIS standard in April 2004. Since many software vendors have worked on the standardization in OASIS, Web Services security has been widely accepted over the past few years with being more interoperable and secure.

Web Services Security (WS-Security) (Lawrence and Kaler, 2006a) is a specification that protects SOAP messages, ensuring end-to-end security for web services by storing all security information as a part of SOAP message. It defines the ways how XML Signature (Bartel M. et al, 2008) and XML Encryption (Imamura T. et al, 2002) are used within SOAP messages to ensure the integrity and confidentiality of SOAP messages. It also defines the ways security tokens (e.g. Username / Password and X.509 certificate tokens) (Lawrence and Kaler, 2006b, 2006c) are attached to SOAP messages. To restrict access to a XML Web Services to authorized users and protect messages from being viewed by unauthorized parties WS-Security standards must be precisely followed.

Interoperability of WS-Security is an important issue. WS-I, Web services interoperability organization, is developing a Basic Security Profile (Chumbley and Martin, 2008) to advance interoperability of WS-Security. On top of the WS-Security protocol there are another six security specifications; WS-Policy (Vedamuthu et al, 2007), WS-Trust (Lawrence and Kaler, 2007a), WS-Privacy (IBM and Microsoft, 2002), WS-SecureConversation (Lawrence and Kaler, 2007b), WS-Federation (BEA Systems et al, 2006) and WS-Authorization (IBM and Microsoft, 2002). WS-Security standardization has taken big steps. Applying WS-Security to real life projects is the next step.

As the web services are widely accepted, the tools that developers use have covered all the needs from security to transaction management to provide reliable communication and integration. Windows Communication Foundation (WCF) (Löwy 2008;Resnick et al, 2009) is a framework that provides standards based functionality to develop and deploy secure web services as well as supporting other distributed technologies with its unified programming model on Microsoft Windows platform. Changes and improvements on current standards are applied rapidly. WCF increases developer productivity by providing high level abstraction that simplifies the usage of communication details and standards. In WCF, Web services technologies defined by the WS-* specifications has been implemented. Currently, WCF supports a large set of the WS-* specifications. The communication between parties that uses WCF on both side can be optimized to provide maximum performance.

Security requirements must be considered at the beginning of the project life cycle. Security is a non-functional requirement and must be separated from functional requirements of the project. The separation of security related functions / services and packaging them as an Application Framework (Chen 2004) improves the reusability, maintainability and extensibility. Once the application framework has been developed, it can be used by the applications in the same domain. The application framework provides configuration information so applications can reach the provided functionality out of box and can be easily incorporated into the application infrastructure of the domain. Organizational standards can be easily applied. Application developers need to focus only on the business requirements. Making changes becomes easier and the code does not necessarily have to be recompiled when security requirements change.

In the e-Business world, there is need for an end-to-end security architecture which can be implemented across organizations and trust boundaries. Since Web Services based development is loosely coupled and the application integration is standard based there is great motivation to develop security framework functionality as Web services. We see growth in adopting Web Services and Web services security into the application infrastructures (Sprott et al, 2003; Nakayama 2005; You et al, 2004) from the beginning of Web Services Security standardization. (IBM and Microsoft, 2002)

Our attempt is to provide a framework which all the security functionality exposed as web services in a particular domain or organization. (Universities, Hospitals, e-government agencies etc.) Different applications in the domain can utilize this framework and share security related information. WSSAF is a layered framework and provides secure communication and access control for web services in its container. All the necessary configurations to secure a web service are set to its configuration store by the WSSAF management console. Developers just add necessary codes and some configuration information to interact with WSSAF. When a web service is hosted, WSSAF will be ready to protect it with the help of a small configuration phase. WSSAF provides security services to secure Web Services in the container by using the same standards used to develop WSSAF.

Within this thesis, principles and the architectural model of the framework is presented. We expect and show that the WSSAF model encourages and shortens the transition period especially for the small businesses to take big steps into the Web Services world. The proposed framework models all the security related functions like authentication, authorization as web services and adds a layer on top of Web Services Security Toolkits. WSSAF focuses on being a lightweight and cost effective SOA security framework, to adopt the SOA vision easily, to provide centralized security for Web services containers and to achieve easy integration through Web Services in business domains like hospitals, universities.

Securing Web Services with message level security (WS-*) is the primary concern. The other models for securing web services and Web Services Security threats are not within the scope of this work. Finally, a prototype is given to show the effectiveness, applicability and feasibility of the framework.

CHAPTER 2

XML AND WEB SERVICES SECURITY

2.1 INTRODUCTION TO XML AND WEB SERVICES

XML (Bray et al, 2008) describes data and behavior in XML documents that are shared between information systems and processed by computer programs. XML Web Services use XML messages to exchange data.

The main design goals of XML are usability over the internet, easiness to write, support for wide variety of applications, easiness in processing by computer programs. XML is recommended by the World Wide Web Consortium (W3C).

Web services (Booth D. et al, 2004) are interoperable distributed software systems which publish standard interfaces and support message based access to help efficient service integration.

Main technologies behind web services are extensible Markup Language (XML) (Bray 2008), Simple Object Access Protocol (SOAP) (Gudgin et al, 2007), Universal Description, Discovery, and Integration (UDDI) (Clement et al, 2004) and Web Services Description Language (WSDL) (Chinnici et al, 2007). XML and SOAP make it possible for web services to communicate with each other and their clients over HTTP (W3C Specification, 2009) protocol. UDDI is used to search for other web services declared within the UDDI directory and WSDL is used to get a description on how to use the published web services.

Figure 2.1 illustrates the logical view of the relationship between Web services roles and operations.

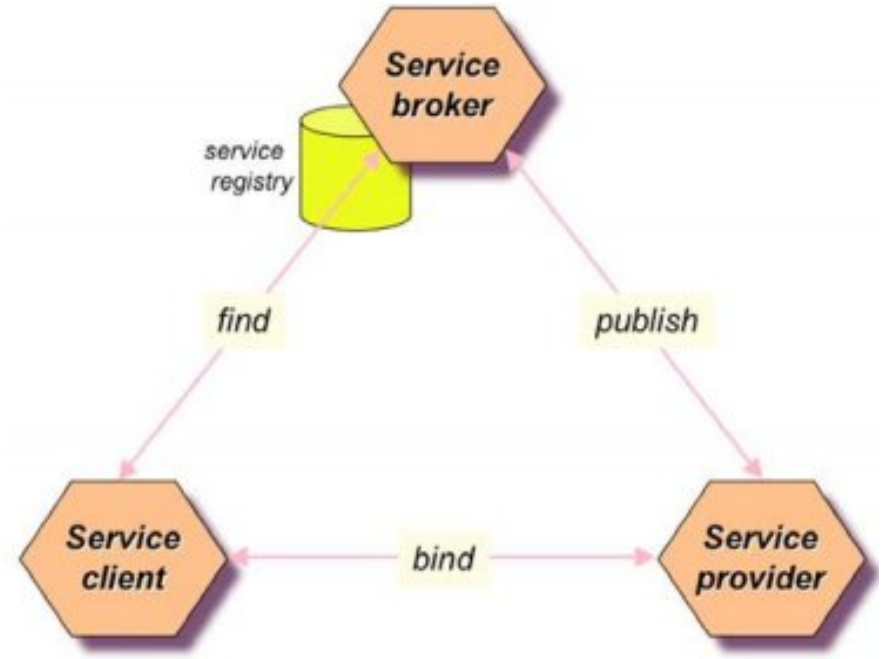

Figure 2.1. Web Service roles and operations (Papazoglou, 2006)

2.2 XML SECURITY

XML is a well known technology and is adapted widely for structuring data, and therefore XML-based security utilizes the advantages of XML to handle complex requirements for security between information systems.

The aim of the XML Security Standards developed by W3C and OASIS is the definition of metadata to protect XML documents and elements. To meet the necessary security requirements the XML Security standards define XML vocabularies and processing rules. Figure 2.2 illustrates these standards. By utilizing these standards, access control, rights management and communication protection can be provided in the development of XML Web Services.

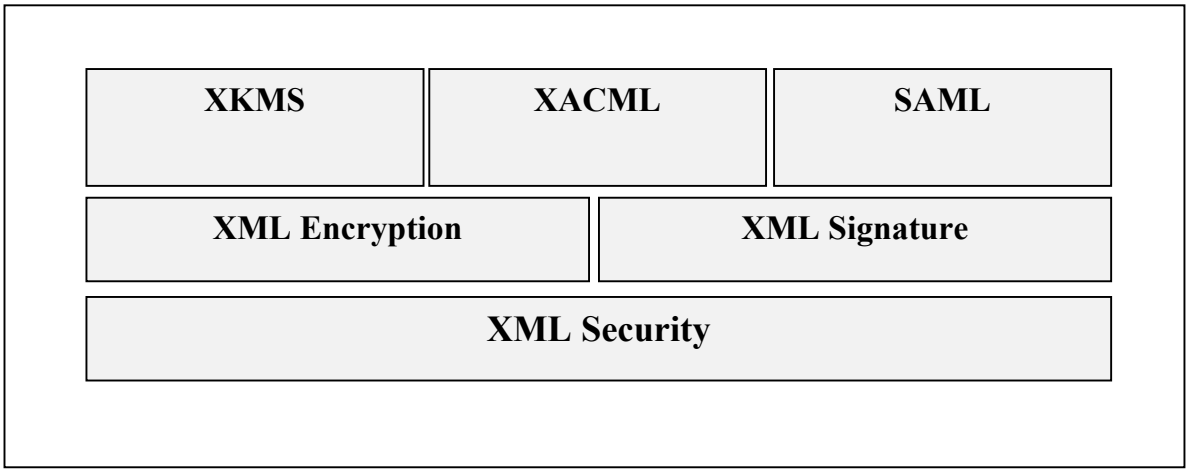

Figure 2.2. XML Security Standarts

In the following subsections, XML Signature, XML Encryption, XML Key Management Specification (XKMS), XML Access Control Markup Language (XACML), and Security Assertion Markup Language (SAML) will be explained.

2.2.1 XML Signature

The XML Signature (Bartel et al, 2008) defines a XML syntax for digital signatures. Digital signatures are used to ensure the recipient that the message was in fact transmitted by the trusted service provider. Digital signatures become an embedded part of the document. It also ensures the integrity of the message transmitted, as well as support for standard non-repudiation.

One of the main advantages of the XML signature over standard digital signatures is that it utilizes the XML power. Message integrity is provided by using the XML Signature, which is developed by W3C.

2.2.2 XML Encryption

The XML Encryption (Imamura et al, 2002), is a specification that defines how to encrypt the contents of an XML element. The specification contains a standard model for encrypting both binary and textual data. Encrypted data and other related information can be expressed in the form of XML elements.

This specification can be used to provide confidentiality between SOAP intermediaries. Any part of a document can be encrypted using different encryption keys. While the message is in transit some elements of the SOAP message are kept confidential from SOAP intermediaries. Message confidentiality is provided by using XML Encryption, which is developed by W3C.

2.2.3 XML Key Management Specification (XKMS)

XML Key Management Specification (XKMS) (Ford et al, 2001), is a protocol which defines XML message formats for public key management. Public Key management supports the operations including public key registration, public key discovery, public key validation and public key revocation.

In its essence the XKMS protocol is designed as a web service in front of the Public Key Infrastructure (PKI). XKMS supports a number of public key infrastructure (PKI) technologies removing the necessity for integrating proprietary PKI products.

XKMS Web service transfers the complexity of the entire public key management from the client to itself. There are a number of advantages to using XKMS as a front-end to PKI. As mentioned above one of these advantages is reduction in client complexity. Clients can access an XKMS compliant server in order to receive updated key information for encryption and authentication. This makes it much easier to add new features without having to update the clients because centrally updating XKMS will be more efficient. So the client deployment problem is also handled by XKMS web services. Another advantage

is shifting the processing load from the client to the web service. By doing this, it is more efficient to publish an enterprise wide security policy for all the clients.

Essentially, the XKMS protocol is a request-response protocol based on SOAP. The XKMS specification defines two web services; the XML Key Registration Service Specification (X-KRSS) and the XML Key Information Service Specification (X-KISS). X-KISS is used for verification purposes of public keys or certificates. X-KRSS is used to register public and private key. XKMS is developed by W3C and designed to be used with the XML Signature and the XML Encryption.

2.2.4 XML Access Control Markup Language (XACML)

XACML is developed by the OASIS. XML Access Control Markup Language (XACML)(Moses 2005) defines an XML vocabulary to specify the rules on which access control decisions are made. The main goal of XACML is to standardize access control language in XML syntax. XACML provides exchanging or sharing authorization policies between different organizations without having to translate the policy for each organization.

By using a rule-combining algorithm, within the policy, the rules can be combined together. XACML formalizes the algorithms used to link rules together to make policies, and the algorithms to link together policies to make meta-policies.

To define a rule in XACML we need the focus on three parts; a target, an effect and a set of conditions. The target is defined as the space of decision requests referring to actions on resources by subjects. An effect can be a ‘permit’ or ‘deny’. To make a rule more dynamic, a set of conditions can be applied to the rule.

Multiple rules can make up a XACML policy. This is one of the most important aspects of the specification. In this way an organization can use a single policy to enforce access to many different resources.

2.2.5 Security Assertion Markup Language (SAML)

Security Assertion Markup Language (SAML)(Cantor 2005), is an XML-based standard for exchanging authentication and authorization data between security domains, which includes an identity provider and a service provider. The identity provider provides local authentication services to the user. The Service provider relies on the identity provider to identify the user. The request is made by the user; the identity provider passes a SAML assertion to the service provider. On the basis of this assertion, the service provider makes an access control decision.

From a user's perspective, after the single sign-on has occurred and a trust relationship has been established, an assertion is produced. Then the assertion about that authentication can be used between trusted security domains. In addition to the authentication statement, assertions can include attributes and authorization statements. Service providers use these statements to make access control decisions.

SAML is developed by OASIS. The final goal of SAML is to provide a 'single-sign-on' solution between different web services.

2.3 WEB SERVICES SECURITY

2.3.1 WS-Security

WS-Security (IBM and Microsoft, 2002) was developed by IBM, Microsoft, and VeriSign. These companies have submitted WS-Security specification to OASIS (OASIS Community), the international standards organization for e-business. OASIS has approved WS-Security as an OASIS standard in April 2004.

WS-Security is a specification that protects SOAP messages and provides end-to-end security for web services. It describes the way how the XML Signature (Bartel M. et al, 2008) and the XML Encryption (Imamura T. et al, 2002) are used within SOAP messages. It also describes the way how security tokens (e.g.Usernames/Password, X.509 certificate and kerberos tokens) (Lawrence and Kaler, 2006b, 2006c, 2006d) are attached to SOAP

messages. WS-Security provides message integrity, message confidentiality and credential transfer with these enhancements in SOAP messages.

Using Web Services provides an abstract layer for integration between information systems running on different platforms. These information systems have different security implementation developed with different security technologies. To solve these requirements, WS-Security also provides an abstract layer for communicating security related data between information systems.

WS-Security protects each message individually and therefore supports more point of interaction, that is, messages must be routed one or more intermediaries to reach to last node. Another benefit of using WS-Security is that some parts of a message can be encrypted. Using this capability can have an impact on performance gain. WS-Security does not require a special transport layer. HTTP (W3C Specification) is most preferred way to transport SOAP messages between nodes. One disadvantage of using WS-Security is that it can actually affects the performance if multiple messages are sent between nodes.

2.3.2 WS-Security Standards

WS-Security allows Web services to apply security to SOAP messages through confidentiality and integrity checks to the whole or part of the message. Web Services Security Specifications (WS-*) is a set of standards based protocols designed to secure web service communication. Figure 2.3 illustrates these standards and their relationship to the WS-Security.

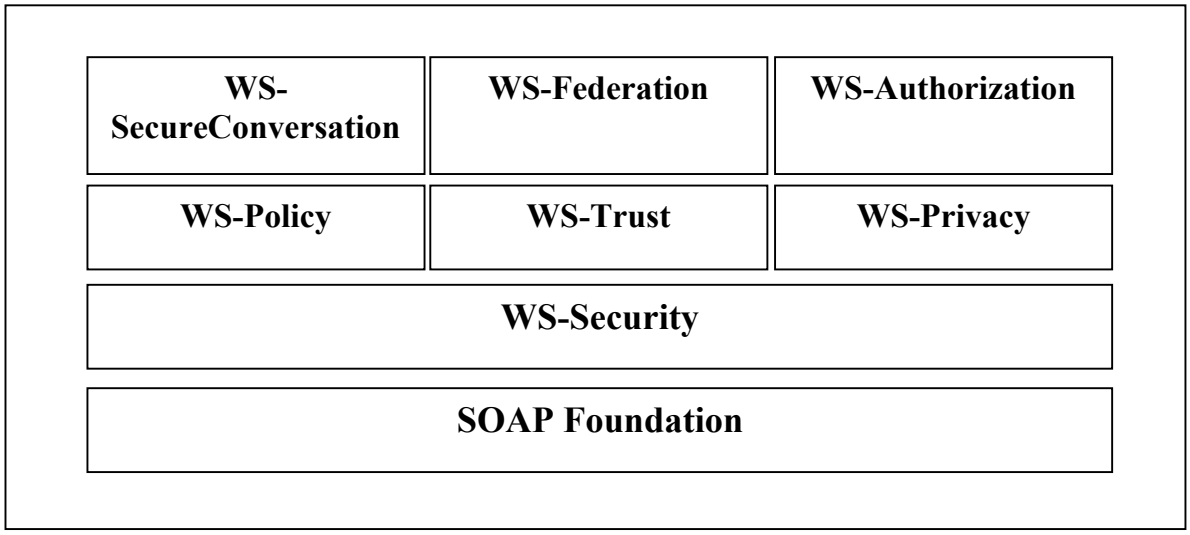

Figure 2.3. WS-Security Standards [IBM and Microsoft, 2002]

On top of the WS-Security protocol there are further six security specifications; WS-Policy (Vedamuthu et al, 2007), WS-Trust (Lawrence and Kaler, 2007a), WS-Privacy (IBM and Microsoft, 2002), WS-SecureConversation (Lawrence and Kaler, 2007b), WS-Federation (BEA Systems et al, 2006) and WS-Authorization (IBM and Microsoft, 2002).

WS-Trust

WS-Trust (Lawrence and Kaler, 2007a) describes the framework for managing, establishing and assessing trust relationships between web services to communicate securely.

WS-Trust allows Web services to use security tokens to establish trust in a brokered security environment. It defines the concept of a Security Token Service. (STS) STS is a web service that issues security tokens as defined in the WS-Security specification. WS-Security will be used to transfer the required security tokens.

WS-Secure Conversation

WS-SecureConversation (Lawrence and Kaler, 2007b) describes a framework to establish and share secure context with associated session keys for parties that want to exchange multiple messages.

With WS-Security every message goes through a checking process against the security policy which can cause performance degradation. There is no session concept for a group of SOAP messages in WS-Security. WS-SecureConversation fills this gap by allowing a client and a web service to authenticate using SOAP messages and establish a security context including authentication information.

WS-SecureConversation builds on top of WS-Policy (Vedamuthu et al, 2007), WS-Security (IBM and Microsoft, 2002), and WS-Trust (Lawrence and Kaler, 2007a) to enable secure communications between client and service.

WS-Policy

WS-Policy (Vedamuthu et al, 2007) allows Web services to define policy requirements for endpoints. These security requirements include the privacy attributes, the supported algorithms for encryption and digital signatures and how this information may be bound to a web service.

WS-Policy is extensible and new types of requirements and capabilities can be described. WS-Policy allows organizations initiating a SOAP exchange to discover what type of security tokens are understood, looks like the way that WSDL describes a web service.

WS-Authorization

WS-Authorization (IBM and Microsoft, 2002) provides a standard mechanism for authorizing the requirement and capability statements to the Web service.

WS-Authorization describes how a web service's access policies are specified and managed. It is designed to be flexible and extensible with respect to both authorization format and authorization language.

WS-Privacy

WS-Privacy (IBM and Microsoft, 2002) describes a model for how Web services and requesters state privacy preferences and organizational privacy practice statements.

WS-Privacy works with WS-Policy, WS-Security and WS-Trust to communicate privacy policies. Organizations state these privacy policies for their web services and the clients conform to these privacy policies to be successful in communication. WS-Privacy explains how privacy requirements can be included in WS-Policy descriptions. WS-Trust is used to check the privacy claims encapsulated within SOAP messages against client preferences and organizational privacy statements.

WS-Federation

WS-Federation (BEA Systems et al, 2006) describes the management and brokering of the trust relationships in a heterogeneous federated environment including support for federated identities.

WS-Security (Lawrence and Kaler, 2006a), WS-Trust (Lawrence and Kaler, 2007a), and WS-SecurityPolicy (Lawrence and Kaler, 2007c) provide a basic model for federation between identity providers and relying parties. These specifications define standards and the way to specify a set of assertions in security tokens to protect and authorize web services requests. WS-Federation extends these standards by adding claim transformation model in security token exchange to set richer trust relationships. In this way, a client in a security domain accesses to protected resources managed in different domain with its own security principle attributes.

CHAPTER 3

THE WSSAF MODEL

3.1 PRINCIPLES OF THE WSSAF MODEL

The main principles that were in consideration during the design of WSSAF can be summarized as follows;

(a) Modeling based on Web Services Architecture and standards.
(b) Using Web Services Security standards (Message Security)
(c) Transition to Service Oriented Architecture (SOA) using Web services.
(d) Being lightweight and easy configurable SOA Security Framework Model.
(e) Providing Security Framework functions as Web Services.
(f) Intercommunication within WSSAF is web services based. (Based on Web services standarts)
(g) Securing Web Services in the WSSAF container with WSSAF Security Web services.
(h) Centralized Security Services and Security information Store (Security Configurations, Certificates, Passwords, Policies, Logs) for E-Business applications.
(i) Abstract Data model for Web Service Security configurations.
(j) Mobility of WSSAF Services & Consolidating with Data Service.
(k) Providing end-to-end security and application layer security.
(l) Helping to follow Organizational Security Standards.

3.2 CONCEPTS

3.2.1 Public Key Encryption and Decryption

Public key encryption, also known as asymmetric encryption, is based on a key pair named as the private key and the public key. There is a mathematical relation between keys. Anyone can use the public key to encrypt a message but the decryption can be done only with the corresponding private key. The sender of message converts the plaintext message into cipher text by encrypting it with the message recipient's public key. The recipient of the message converts the cipher text back into the plaintext message by decrypting it with the corresponding private key. Figure 3.1 illustrates the process.

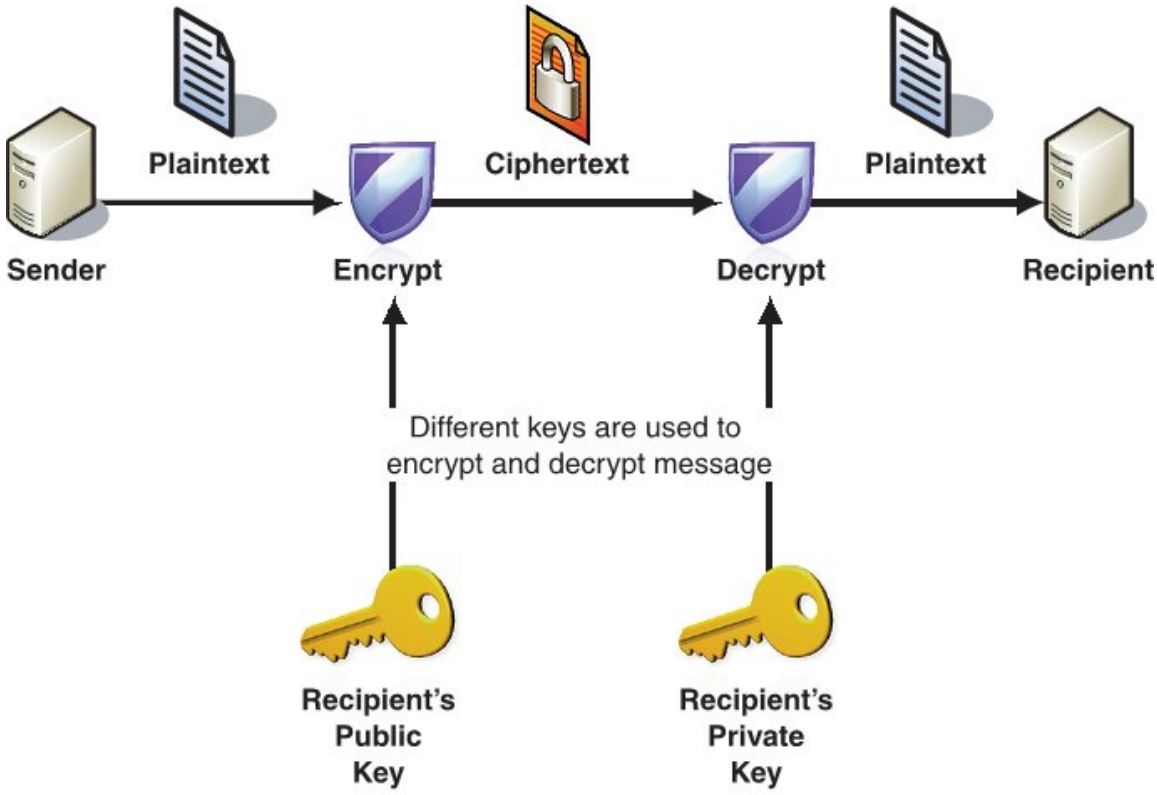

Figure 3.1. Public key encryption and decryption (Microsoft Patterns And Practices Group, 2005)

Public key encryption, can assure the message sender that only the recipient will be able to read the message.

3.2.2 X.509 certificates

The X.509 specification (Adams and Farrell, 1999) is part of the X.500 series of recommendations that define a directory structure. The directory may serve as a storage of public key certificates which is named as a Public Key Infrastructure (PKI). The public keys are maintained in X.509 certificates. Each certificate contains a certain users information including the public key and is signed with the private key of a trusted third party known as a certificate authority (CA).

The X.509 standard which defines a certificate structure and authentication protocols is broadly accepted. PKI, using X.509 certificates, is capable of establish a bases of trust which can be shared among several organizations. X.509 certificates can be used to provide message confidentiality through encryption or to provide authentication and message integrity through digital signatures. The private key of the sender is used to encrypt the fixed size digest of the message. The encrypted digest is attached to the original message serving as a digital signature. The receiver can verify the signature using the received message and public key of the sender which can be obtained from the X.509 certificate. This process is illustrated in Figure 3.2.

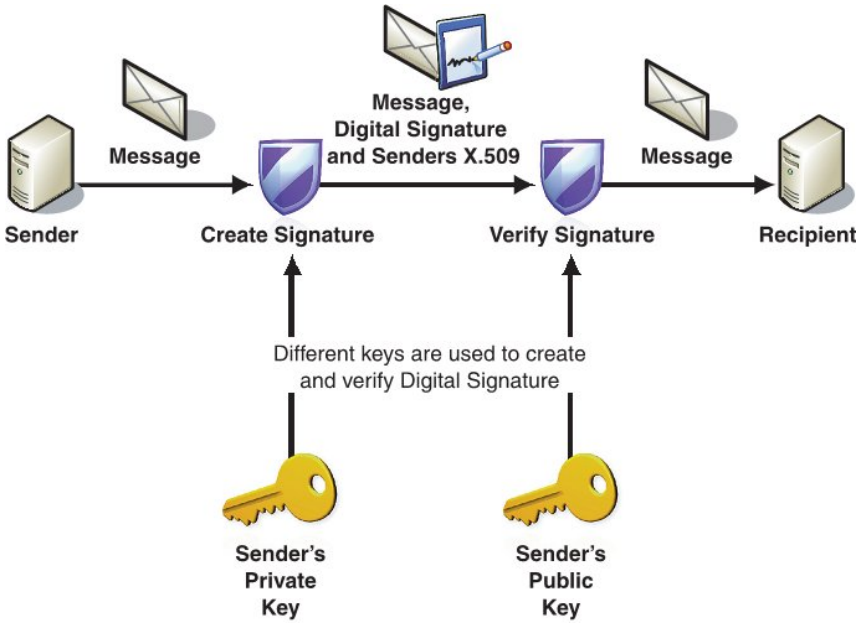

Figure 3.2. Process of using public keys to sign a message (Microsoft Patterns And Practices Group, 2005)

X.509 certificates contain several required and optional attributes that enable the identification of the subject. We can get the following list of attributes in a X.509 V3 certificate:

Version number: The version of the certificate.

Serial number: A unique identification number for the certificate.

Signature algorithm ID: The algorithm used to generate the digital signature.

Issuer name: The name of the certificate issuer.

Validity period: The time period that the certificate is valid.

Subject name: The name of the subject that certificate represents.

Subject public key information: The public key algorithm.

Issuer unique identifier: The identifier for the issuer.

Subject unique identifier: The identifier for the subject.

Extensions: Extensions that can be used to store additional information.

Signed digest of the certificate data: The digest of the all above fields encrypted using the issuer's private key. (Digital signature)

3.2.3 WS-Security X.509 Binary Security Token

X.509 certificates can be used at the message layer as binary security tokens in accordance with the WS-Security specification to sign and encrypt messages. The benefits of using X.509 at the message layer with binary security tokens include:

(a) Flexibility to provide point-to-point or end-to-end security. This allows messages to be persisted in a secure state for short periods or for longer periods.
(b) A high degree of interoperability. The messages sent over the wire are in accordance with standards.

Message layer security also has the following shortcomings:

(a) Because the message layer is further away from the hardware layer, processing message layer security with X.509 certificates tends to have a greater impact on system performance than implementations that are lower in the protocol stack.
(b) Message layer security that uses X.509 tends to be more complex to implement than security that uses X.509 certificates at other layers.

3.2.4 Participants in Authentication process

Brokered authentication with STS involves the following participants:

(a) **Certificate authority (CA).** A CA is an authentication broker that is responsible for authenticating clients and issuing valid X.509 certificates.
(b) **Certificate store.** Store where the X.509 certificates are located.
(c) **Client.** The client that sends request messages to the Web service and get the response messages. The client provides the credentials for authentication.
(d) **STS.** The STS is the Web service that authenticates clients by validating credentials that are sent by the client. The STS can issue a security token to the client to prove that it is authenticated.
(e) **Service.** The web service that requires authentication and authorization of a client to give access its resources.

3.2.5 Security Token Service (STS)

STS is a web service that issues security tokens as defined in the WS-Security specification. For all the web services in a business domain can utilize from STS as a common access control infrastructure. Since every web service needs to authenticate the clients, STS negotiates trust between clients and Web services. STS is trusted by both of the client and the web service.

The client sends an authentication request with necessary credentials to the STS. The STS validates the credentials. STS issues a security token that provides proof that the client has authenticated with the STS. The client gets the security token from STS. The client sends a request including the signed security token to the Web service. The Web service validates that the token was issued by a trusted STS. This means that the client has successfully authenticated with the STS.

The specification used for issuing security tokens is based on WS-Trust (Lawrence and Kaler, 2007a). WS-Trust allows Web services to use security tokens to establish trust in a brokered security environment. WS-Trust is a Web service specification that builds on WS-Security (Lawrence and Kaler, 2006a). WS-Trust describes the details for issuing, exchanging and validating security tokens.

In WS-Trust specification, the message sent to the STS to request a security token is known as a Request Security Token (RST) message. The RST message contains credentials for the client to be authenticated. The response message sent from the STS to the client is known as a Request Security Token Response (RSTR) message. The RSTR contains a security token.

Figure 3.3 illustrates the authentication process between the client, the service and the STS.

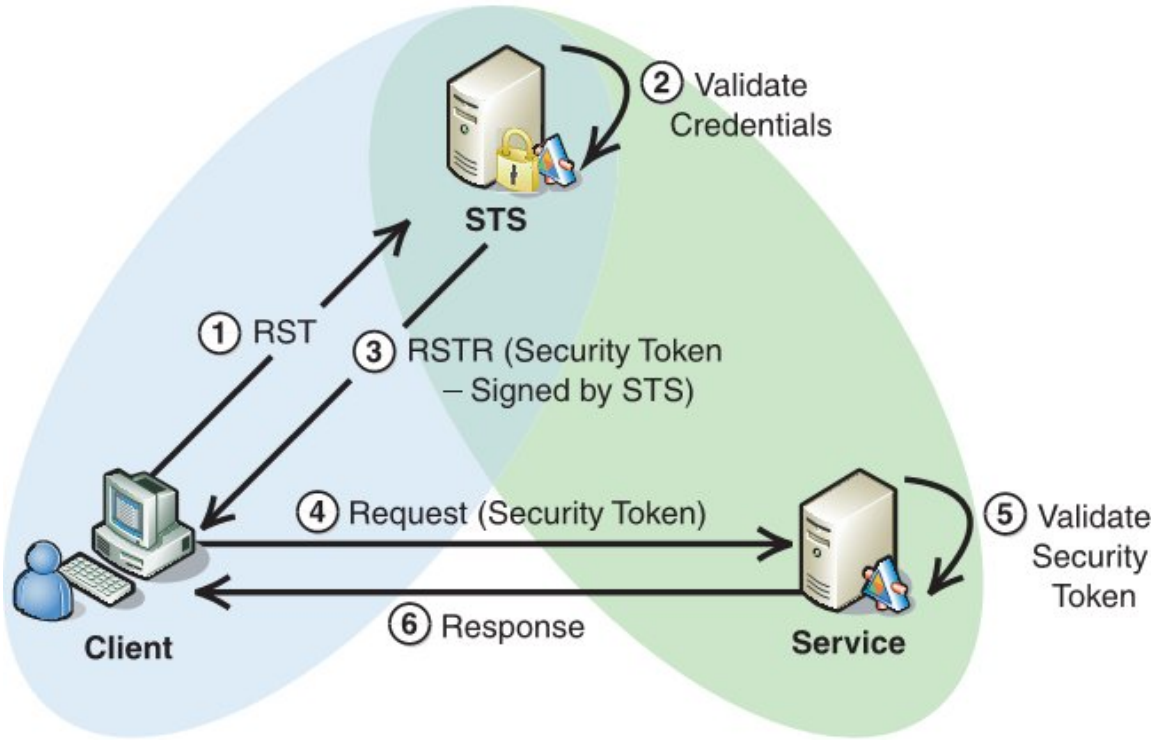

Figure 3.3. Security Token Service Issuance and Request/Response (Microsoft Patterns And Practices Group, 2005)

In this process, the following steps are executed;

1. The client prepares and sends authentication request to the STS.

2. The STS validates the client's credentials.

If the client's credentials are valid, check for the authorization policies for issuing tokens. There may be policies to control issuing tokens for users which has a specific role or for valid X.509 certificates etc.

3. The STS issues a security token to the client.

If the client's credentials are validated, the STS issues a security token in an RSTR message to the client. The security token is signed by the STS.

4. The client prepares and sends a request message to the service.

The request message includes the issued security token.

5. The service validates the security token and processes the request.

The security token is validated by the service to verify that the token was issued by the trusted STS. After the token is validated by the service, it is used to establish security context for the client. Then, the service can make other operations like checking authorization policies or starting auditing activity.

6. The service prepares and sends a response message to the client.

The response message contains result to the request.

3.3 WSSAF CONCEPTUAL MODEL

WSSAF is a Security Application Framework for securing Web services in its container. It utilizes the Windows Communication Foundation (WCF) which supports current Web Services and Web Services Security standards.

Figure 3.4 illustrates the WSSAF position in the Web Services Security stack.

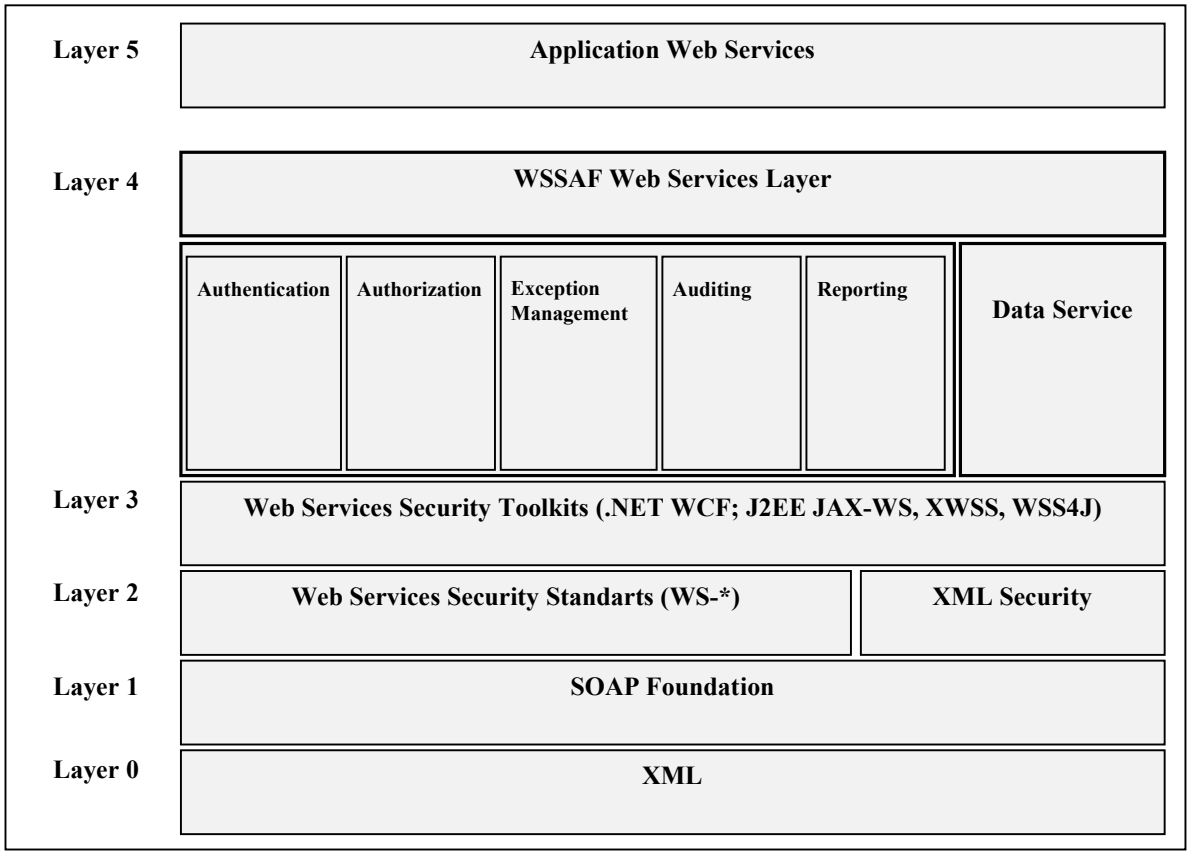

Figure 3.4. WSSAF position in Web Services Security Stack

3.4 WSSAF ARCHITECTURAL MODEL

The WSSAF security platform has three related domains: (1) WSSAF Services, (2) WSSAF container, (3) WSSAF clients.

Figure 3.5 illustrates the architecture of WSSAF and the interaction between related domains.

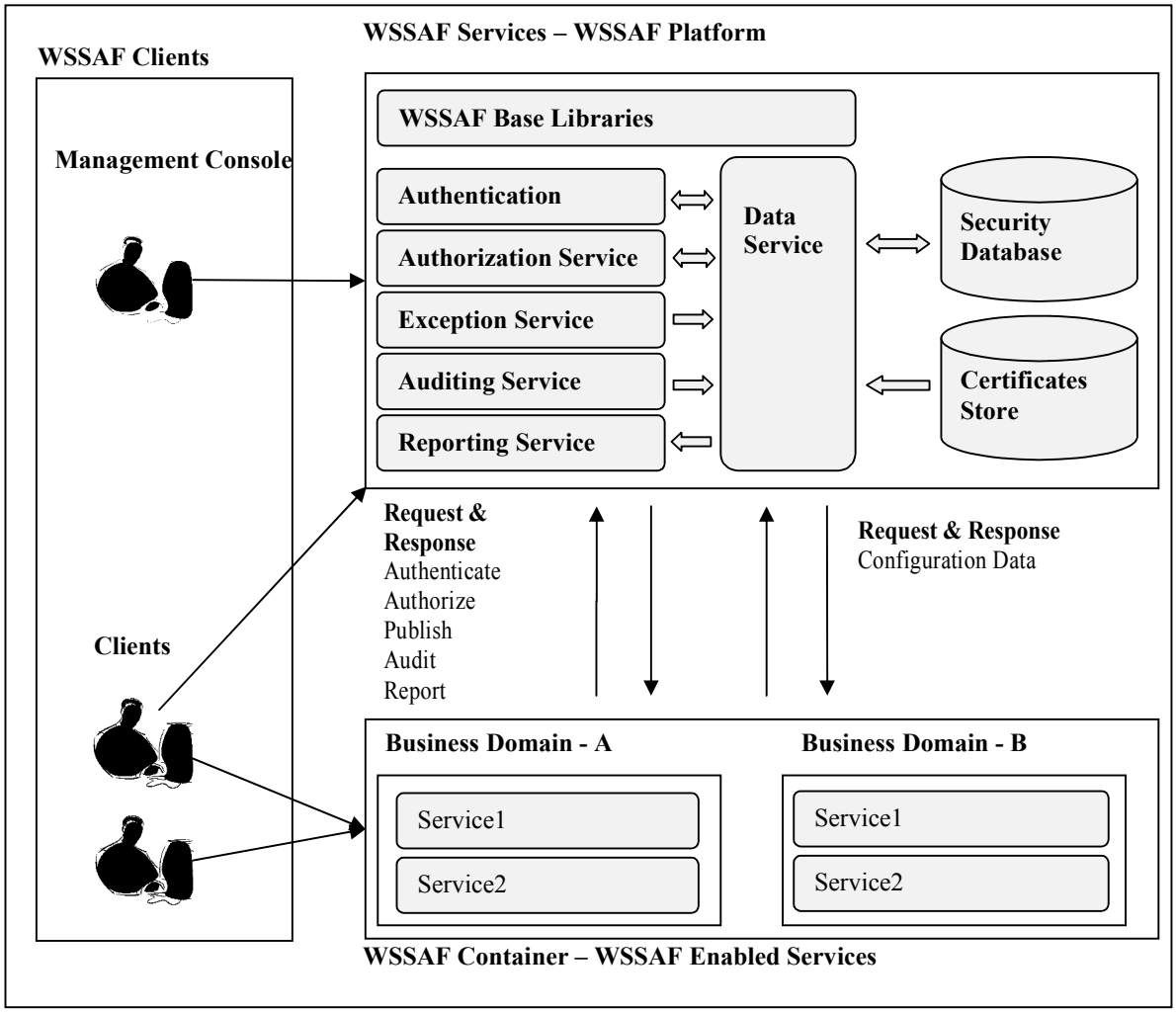

Figure 3.5. WSSAF Architectural Model – WSSAF Security Platform

The WSSAF security platform and the components in the related domains will be explained in the following sub-sections.

3.4.1 WSSAF Services Layer

WSSAF Authentication Service

WSSAF Clients are authenticated whenever a request is made to a service in the WSSAF container. WSSAF supports Username-Password and X.509 (Adams and Farrell, 1999) certificate authentication. Username and passwords are stored in WSSAF security

database. The client certificates are managed by certificate utilities in Microsoft Windows operating system.

Each client is required to provide a credential (password or certificate) to the Authentication Service. If the credentials are validated, the service authenticates the client.

WSSAF Authorization Service

The WSSAF Authorization Service determines whether or not a user can access a resource based on user, group or role information. Authorization policies can be assigned to the specified resources to control access. A resource has no protection without policies.

Web services and Web Service methods can be resources to be protected.

WSSAF Auditing Service

WSSAF Auditing Service is used to collect all important security related events and detailed event information from other WSSAF Services. According to the auditing configurations, these events and detailed event information are persisted into WSSAF store for analyzing and reporting.

As an example, the WSSAF Authentication Service sends the login request and login result to the WSSAF Auditing service to determine the successful logins.

WSSAF Exception Service

WSSAF Exception service is used to collect exceptions and detailed exception information from other WSSAF Services. According to the exception configurations, exceptions and detailed exception information are persisted into WSSAF store for analyzing and reporting.

Handled and unhandled exceptions are caught within the service execution and sent to WSSAF Exception service.

WSSAF Reporting Service

WSSAF Reporting Service has predefined forms of reports for analyzing security related information collected by Auditing and Exceptions services.

For each predefined report, Reporting Service calls the Data Service to fetch data from database. Report is prepared and sent to the requestor in the form of HTML.

3.4.2 WSSAF Data Service

The WSSAF Data Service is a gateway between the WSSAF services and the WSSAF Store. It has domain specific methods that all the services required to fetch from the WSSAF Store and set security related information into the WSSAF Store.

The WSSAF management console uses Data Service directly to browse the configurations and add new configurations or change configurations in the WSSAF Store. WSSAF Configuration Management requirement has been solved by the WSSAF Management Console and the WSSAF Store pair.

All WSSAF services can access Data Service securely. They can be distributed, that is, installed in the same domain or anywhere on internet. The data service also acts as a consolidation service, meaning that information produced by WSSAF services can be persisted into the WSSAF Store.

Figure 3.6 illustrates the component architecture of WSSAF Data Service.

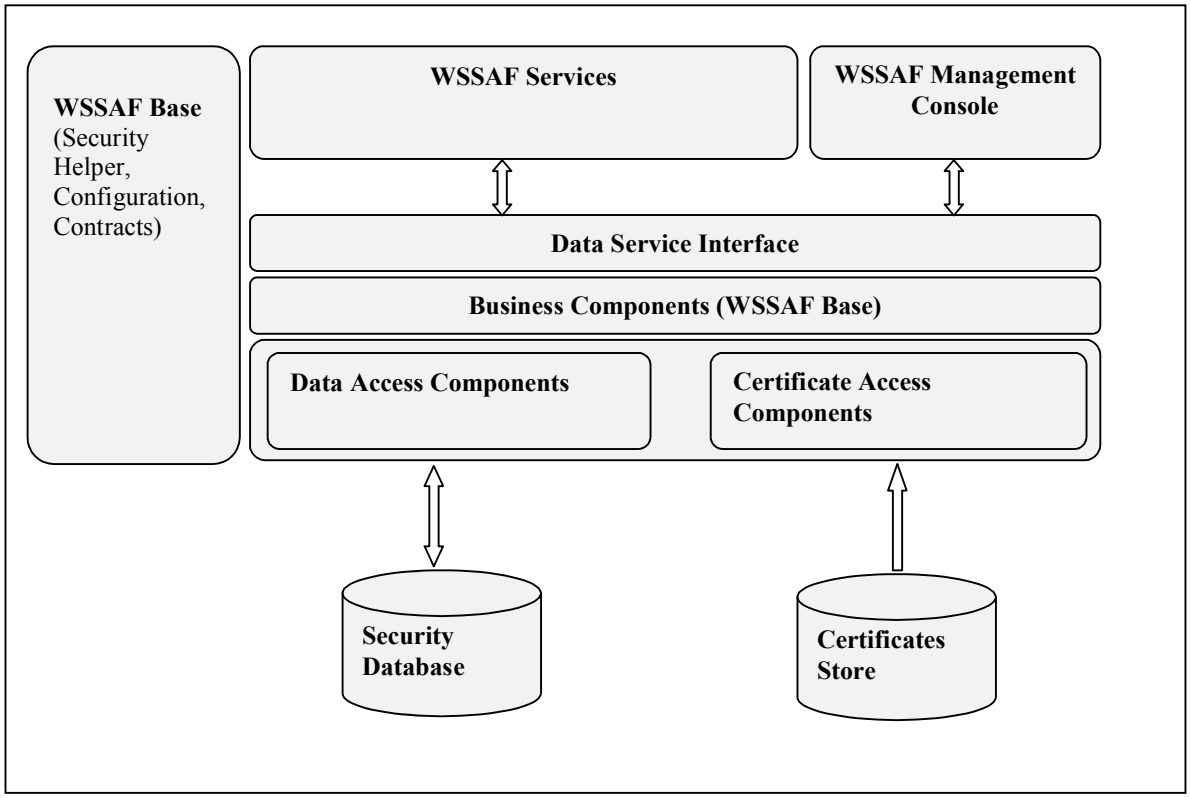

Figure 3.6 WSSAF Data Service Component Architecture

3.4.3 WSSAF Store

The WSSAF Data store has two main parts. One is named as the Security Database which is a relational database model storing the security related information. (Framework Configurations, Web Service configurations, Logs, Policies, Passwords, Exceptions) The other one is called as the Certificates Store. In our case the certificate store is not a separate database. The certificates are stored and managed by Microsoft Windows operating system utilities.

3.4.4 WSSAF Base Libraries

The WSSAF Base libraries contain shared methods which can be referenced by all WSSAF services. The WSSAF Base libraries are primarily used by the WSSAF Data Service which is the base service in the framework. Business components, Data Access components for the Data Service are included in these libraries.

Security helpers, data contracts and special information structures like constants and enumerations are also included in these libraries.

3.4.5 WSSAF Management Console

The WSSAF Management Console is a web application to configure and manage information about users, services and certificates in the framework. It communicates directly with the WSSAF Data Service to accomplish all the tasks assigned to it.

3.4.6 WSSAF Enabled Services

Web Services are defined in the WSSAF container by using the WSSAF Management console. The required service certificates are installed with the utilities provided by Microsoft Windows operating system.

The steps required to develop WSSAF enabled Services are as follows;

1. Develop the service with all the business functionality,
2. Choose hosting options for the services,
3. Configure the service to access WSSAF Data Service,
4. Add WSSAF Data Service access code in startup section of the application hosting the service to get the service configurations dynamically,
5. Use WCF and WSSAF utilities to configure the service in code with dynamic configuration data,
6. Open the service host environment.

The steps that should be followed to make WSSAF enabled services accessible for the clients;

1. Hosting environment calls the service initialization logic,
2. Executing WSSAF access code to get service configurations,
3. Executing WSSAF utilities to get service dynamically up.

3.4.7 WSSAF Clients

There are two types of clients. The WSSAF Management Console is one of the clients. The other clients are the users of WSSAF Enabled Services. Since WSSAF Clients are the users of the framework, they must get the user accounts or provide certificates to obtain secure access to the services in the WSSAF container.

3.5 DISCUSSION ON ARCHITECTURE OF FRAMEWORK MODEL

There are two different architectural models that we have considered during the study:

(a) Framework Services model in a secure domain

In this model, all the framework services are hosted in a secure domain. Framework Services directly communicate with the WSSAF Store through WSSAF base libraries. Framework services are not distributable.

(b) Distributable Framework Services Model

In this model, all framework services are distributable. They can be hosted in the same domain or anywhere on internet. Framework Services communicate with the Data Service to access the WSSAF Store.

We have chosen the second model which has a less performance comparable to the first model due to additional service call to WSSAF Data Service. But the second model has advantages over the first model in terms of utilizing the power of Web Services and easily reaching the goal of modeling Distributed Security Information System. The advantages can be listed as follows;

1. Provides Web Services based Data Access.
2. Easily distribute services in subdomains and consolidate with Data service.
3. Configure additional Data Services to be highly scalable.

CHAPTER 4

WSSAF IMPLEMENTATION

4.1 TECHNOLOGY PLATFORM

WSSAF Web Services were developed with the C# programming language (Microsoft C#) and Windows Communication Foundation (WCF) (Löwy 2008; Resnick et al, 2009) toolkit with Visual Studio 2008 (Microsoft VS, 2008) in .NET platform (Microsoft .NET).

Microsoft SQL Server 2005 Database System (Microsoft SQL, 2005) was used to store security related information. Certificate utilities in Microsoft Windows operating system were used to store and manage certificates. .NET framework utilities were used to get certificates from the certificate store.

The WSSAF Management Console (WSSAF.Web) was developed with ASP.NET (Microsoft ASP.NET) and C#.

4.2 WSSAF SOLUTION ARCHITECTURE

Figure 4.1 illustrates the solution file with related projects.

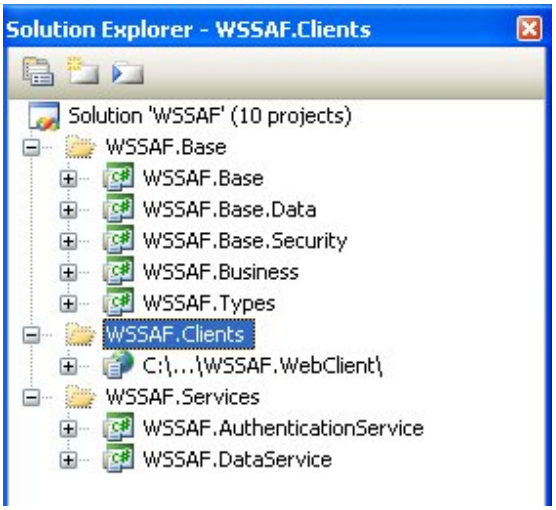

Figure 4.1. WSSAF Solution Architecture

WSSAF Solution Architecture has three groups of projects;

(a) WSSAF Base Projects

WSSAF.Base Project : Base framework types and classes for Constants,Enums,Configuration Management.
WSSAF.Base.Data Project : Data Access Layer and Helper classes.
WSSAF.Base.Security Project : SecurityToken and Security Helper classes.
WSSAF.Business Project : Business classes used by Data Service. (Business Layer of Data Service)
WSSAF.Types Project : Data Contracts and Service Interface types.

(b) WSSAF Service Projects

WSSAF.AuthenticationService Project : Authentication Service implementation. (Service Methods)
WSSAF.DataService Project : Data Service implementation. (Service Methods)

There will be a project for each service. Each Service will reference WSSAF Base Projects for shared types and functionality.

(c) WSSAF Client Projects

WSSAF.WebClient Project : ASP.NET Web Site for WSSAF Management Tasks.

4.3 CONFIGURING WSSAF SERVICES

Web Service configurations are implemented both with configuration files and code. Configuring the services with a configuration file gives us the flexibility of providing and changing configuration data during the development life cycle without rebuilds and re-deployments. The main disadvantage is that it is not type safe and configuration errors will be encountered at run-time. We have chosen to configure the WSSAF Services using configuration files.

WSSAF Services are configured by using the .NET Framework configuration technology. If services are hosted in an Internet Information Services (IIS) site, configurations are added to the Web.config file. Otherwise, the application configuration file is used. By setting configurations for services, a big part of service development work is completed. Main sections in a service configuration file are as follows;

```
<system.ServiceModel>
  <services>
    <service>
      <endpoint/>
    </service>
  </services>
   <client>
   </client>
  <bindings>
    <binding>
    </binding>
  </bindings>
  <behaviors>
      <behavior>
     </behavior>
  </behaviors>
</system.ServiceModel>
```

<system.ServiceModel> node can contain “services”, “bindings”, “behaviors”, “clients” sub-nodes.

<Services> Element

The services element is a container for all services the application hosts.

<Service> Element

Each service has “Name” and “BehaviourConfiguration” attributes.

“Name” Attribute: Specifies the type that provides an implementation of a service contract.

“BehaviourConfiguration” Attribute: Specifies the name of one of the behavior elements found in the behaviors element.

One or more endpoints can be defined in each service.

<Endpoint> Element

Each endpoint has “address”, “binding”, “BindingConfiguration”,“contract” attributes.

“Address” Attribute: Specifies the service's Uniform Resource Identifier (URI). The URI can be an absolute address or an address that is given relative to the base address of the service. An empty string value indicates that the endpoint is available at the specified base address.

“Binding” Attribute: Typically specifies a system-provided binding like WsHttpBinding, or a user-defined binding. Bindings are used to specify the type of transport, encoding, and protocol details required for clients and services to establish a communication. The WSHttpBinding uses the HTTP transport and provides message security. It also provides more web services features like transaction support, reliable messaging, and addressing.

“BindingConfiguration” Attribute: The default values of a binding can be modified by configuring the appropriate binding element in the bindings element. This attribute should be given the same value as the name attribute of the binding element that is used to change the defaults.

“Contract” Attribute: Specifies the interface that defines the contract. This is the interface implemented with the type specified by the name attribute of the service element.

<Client> Element

This element contains a list of endpoints that the client uses to connect to the services. Each endpoint included in the client section defines its own binding, behavior, and contract.

<Bindings> Element

The bindings element contains the configurations for all bindings that can be used by any endpoint defined in any service. Each binding specifies the transport, encoding and protocols included in the communication between client and service.

<Binding> Element

The binding element contained in the bindings element can be either one of the system-provided bindings or a custom binding. The binding element has a name attribute that may be used in the bindingConfiguration attribute of any endpoint element of a service.

We can use the binding element to configure different types of predefined bindings provided by Windows Communication Foundation (WCF). It is not possible to add new elements or attributes to a system-provided binding. We can develop a custom binding for our specific requirements.

<Behaviors> Element

The behaviours element contains the configurations for all behavior elements that define the behaviors for a service. Behaviors are WCF built-in classes that affect runtime execution flow.

<Behavior> Element

Each behavior element is described by a name attribute and provides either a built-in system behavior or a custom behavior. The behavior element contains a collection of configurations for the behavior of a service. Each behavior is accessed by its name and services use the behaviors' name in its behavior configuration attribute.

4.4 WSSAF SERVICES

In this section, the WSSAF Service configurations and functionalities will be explained.

4.4.1 Authentication Service

The WSSAF Authentication Service is a Security Token Service (STS) that issues security tokens for authentication. The WSSAF clients which try to access a WSSAF Enabled Web Service send an authentication request, with necessary credentials, to the WSSAF Authentication service. The WSSAF Authentication Service verifies the credentials presented by the client, and then as a response, it issues a security token that provides a valid result that the client has authenticated with WSSAF Authentication Service or throws an exception. The client presents the security token to the WSSAF Enabled Web service. The Web service verifies that the token was issued by the WSSAF Authentication Service, which proves that the client has successfully authenticated with the WSSAF Authentication Service.

The Authentication Service has an Issue method for accepting the message coming from client, process the message and prepare a response message. The Authentication Service supports two types authentication for clients. Username authentication and X.509 certificate authentication.

For these authentication types, there are two classes called as "CustomUserNameValidator" and "CustomX509CertificateValidator" with Validate methods. These class names are set with attributes

customUserNamePasswordValidatorType and customCertificateValidatorType in the configuration file. WCF runtime calls validate method of the specified classes to authenticate the client with custom code.

The Authentication Service was configured fully in the application configuration file. The Authentication Service configuration file is as follows;

<Services> Element of Authentication Service configuration file

```
<services>
        <service behaviorConfiguration="STSBehaviour"
        name="WSSAF.Authentication.AuthenticationService">
      <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange" />
      <endpoint address="" binding="wsHttpBinding" bindingConfiguration="stsBinding"
contract="WSSAF.Authentication.IAuthenticationService" />
      <endpoint address="Certificate" binding="wsHttpBinding"
bindingConfiguration="stsBindingWithCertificate"
contract="WSSAF.Authentication.IAuthenticationService" />
      <host>
        <baseAddresses>
            <add baseAddress="http://localhost:8000/AuthenticationService/"/>
        </baseAddresses>
        <timeouts openTimeout="00:10:00" />
      </host>
    </service>

 </services>
```

There is one service element with the name “WSSAF.Authentication. AuthenticationService”. The behavior configuration name is “STSBehaviour”. There is a behavior element with the name “STSBehaviour” in the behaviors section. There are 3 endpoints for the service:

1. Metadata publishing endpoint.
2. Username authentication endpoint.
3. Certificate authentication endpoint.

Address, binding and binding configurations are specified for each endpoint. There is a binding element in bindings section for each binding configuration used in endpoints. For MetaDataExhange endpoint, there is no binding configuration in the bindings section. For the binding configuration used in this endpoint, system defaults are used. The interface implemented by the service is given in the contract attribute. The BaseAddress element specifies the base addresses used by the service. The "address" attributes can be used relative to the BaseAddress element like in this case.

<Client> Element of Authentication Service configuration file

```
<client>
    <endpoint name="DataService" behaviorConfiguration="DataServiceBehavior"
            address="http://localhost:8020/DataService/"
            binding="wsHttpBinding"
            bindingConfiguration="DataService"
            contract="WSSAF.Types.IDataService">
            <identity>
              <dns value="DS" />
            </identity>
    </endpoint>
</client>
```

The Authentication Service is a client of the WSSAF Data Service. An endpoint is defined to access the Data Service with address, binding, binding configuration and contract attributes. Behavior and binding configurations are included in the behaviors and bindings sections. The interface implemented by the WSSAF Data Service is given in the contract attribute.

The "Identity" element specifies settings that enable the client to authenticate the server, when using an issued token. The identity element allows a client to specify the expected identity of the service. It includes the DNS of Data Service X.509 certificate used to authenticate the service. In the handshake process between the client and service, the Windows Communication Foundation (WCF) will ensure that the identity of the expected service matches the values of this element.

<Behaviours> Element of Authentication Service configuration file

```
<behaviors>
    <serviceBehaviors>
```

```
      <behavior name="STSBehaviour">
        <serviceCredentials>
          <clientCertificate>
            <authentication certificateValidationMode="Custom"
customCertificateValidatorType
="WSSAF.Authentication.AuthenticationService+CustomX509CertificateValidator,
WSSAF.AuthenticationService" />
          </clientCertificate>
          <userNameAuthentication userNamePasswordValidationMode="Custom"
customUserNamePasswordValidatorType="WSSAF.Authentication.AuthenticationService
+CustomUserNameValidator, WSSAF.AuthenticationService"/>
          <se rviceCertificate storeLocation="LocalMachine"
                storeName="My"
                x509FindType="FindBySubjectDistinguishedName"
                findValue="CN=STS" />
        </serviceCredentials>
        <serviceDebug includeExceptionDetailInFaults="true" />
        <serviceMetadata httpGetEnabled="true" />
      </behavior>
    </serviceBehaviors>
    <endpointBehaviors>
      <behavior name="DataServiceBehavior"  >
        <clientCredentials supportInteractive="false" >
          <serviceCertificate>
           <authentication certificateValidationMode="PeerOrChainTrust"/>
           <defaultCertificate storeLocation="CurrentUser"
                                        storeName="TrustedPeople"
                                        x509FindType="FindBySubjectDistinguishedName"
                    findValue="CN=DS"/>
          </serviceCertificate>
        </clientCredentials>
      </behavior>
    </endpointBehaviors>

  </behaviors>
```

There are two behaviour elements in this section:

(a) STS behavior

This behavior is the main behavior of the service. The Authentication Service supports two types authentication for clients. Certificate authentication and Username authentication are set to “custom” and class type names are given as “WSSAF.Authentication. AuthenticationService + CustomX509CertificateValidator” and “WSSAF.Authentication.

AuthenticationService + CustomUserNameValidator" with namespace. WCF runtime calls the validate method of the specified classes to authenticate the client with custom code. The Authentication Service certificate is set with attributes like location and key value. "includeExceptionDetailInFaults" attribute in "Servicedebug" element specifies whether to include managed exception information in the detail of SOAP faults returned to the client for debugging purposes. The default is false. "httpGetEnabled" attribute in "ServiceMetaData" element specifies whether to publish service metadata for retrieval using an HTTP/Get request. The default is false.

(b) Dataservice behavior

Authentication Service is configured to use Data Service as a client. This behavior is used in "client" configuration section.

Data Service certificate are set with attributes like location and key value.

<Bindings> Element of Authentication Service configuration file

```
<bindings>
    <wsHttpBinding>
      <binding name="DataService" openTimeout="00:10:00" sendTimeout="00:10:00"
allowCookies="false">
        <security mode="Message">
          <message clientCredentialType="UserName"/>
        </security>
      </binding>
      <binding name="stsBinding">
        <security mode="Message">
          <message clientCredentialType="UserName"/>
        </security>
      </binding>
      <binding name="stsBindingWithCertificate">
        <security mode="Message">
          <message clientCredentialType="Certificate"/>
        </security>
      </binding>
    </wsHttpBinding>
  </bindings>
```

There are 3 binding element each of system type WSHttpBinding.

(1) Data service binding

This binding configuration is used in “client” configuration section. Authentication service is a client for the data service. Message security as security mode and Username as credential type are set.

(2) STSBinding for UserName Authentication

This binding configuration is used in a service endpoint. Message security as security mode and Username as credential type are set.

(3) STS Binding for Certificate authentication

This binding configuration is used in a service endpoint. Message security as security mode and Username as credential type are set.

The Authentication Service authenticates the clients using username and certificate by issuing a token that contains information provided by the client. As part of the issued token, Authentication Service signs the token using the private key associated with its certificate. (CN=STS) In addition, it gets a symmetric key and encrypts it using the public key associated with the service certificate. (CN=localhost). When returning the issued token to the client, Authentication Service also returns the symmetric key. The client presents the issued token to the WSSAF Enabled Service, and proves that it knows the symmetric key by signing the message with that key.

4.4.2 Data Service

The WSSAF Data Service is a gateway between WSSAF services and the WSSAF Store. The Data Service is a class that implements the IDataService interface. The Data Service class has three types of methods to access the WSSAF Store through business components. Figure 4.2 illustrates the methods in the DataService class:

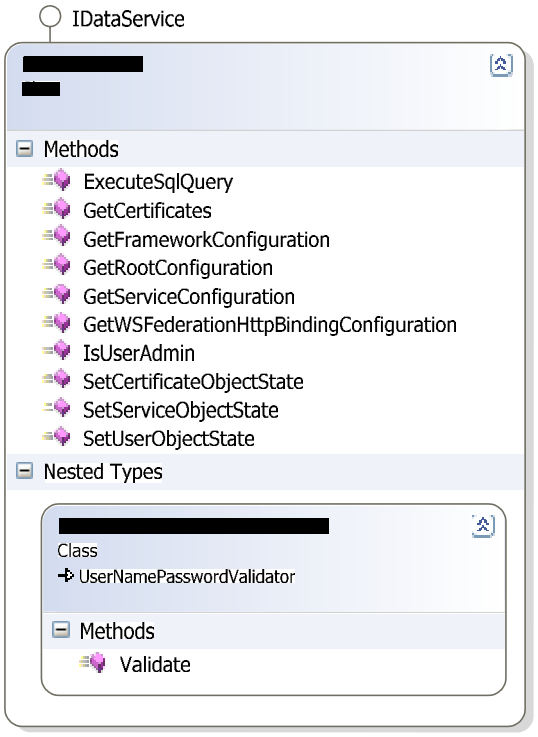

Figure 4.2. Data Service class and methods.

A short method explanation for each methods in the groups are as follows :

(a) Generic Data Access methods

ExecuteSQLQuery : Especially for reporting purposes, It returns reporting data.

(b) Security Model Access Methods

GetFrameworkConfiguration : Gets the configuration parameters like adress, contract, type for all framework services.
GetRootConfiguration : Gets the related configurations at the same time. (RootContract)
GetServiceConfiguration : Gets the configuration parameters for a service defined by a key.
GetWSFederationHttpBindingConfiguration : Gets the federation configuration parameters like security mode,token type, key type.
IsUserAdmin : Checks for the user has administrator rights for WSSAF Management Console.
SetCertificateObjectState : Insert, Update, Delete Certificate object. (Used by WSSAF Management Console)

SetServiceObjectState : Insert, Update, Delete service object. (Used by WSSAF Management Console)
SetUserObjectState : Insert, Update, Delete user object. (Used by WSSAF Management Console)

(c) Certificate Access Methods

GetCertificates : Gets the certificates for Authentication Service and Service utilizes the framework. Certificates with their contents are transferred as raw data in data contracts. The service requesting the certificate validate and convert the raw data to certificate data type by using WSSAF base utilities.

Data Service configuration file looks as follows;

<Services> Element of Data Service configuration file

```
<services>
    <service behaviorConfiguration="DataServiceBehavior"
name="WSSAF.Data.DataService">
      <endpoint address="" binding="wsHttpBinding"
bindingConfiguration="DataService"
        contract="WSSAF.Types.IDataService" />
      <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange" />
      <host>
        <baseAddresses>
          <add baseAddress="http://localhost:8020/DataService/" />
        </baseAddresses>
        <timeouts openTimeout="00:10:00" />
      </host>
    </service>
</services>
```

There is one service element with the name “WSSAF.Data.DataService”. The behavior configuration name is “DataServiceBehavior”. There is a behavior element with the name “DataServiceBehavior” in the behaviors section. There are 2 endpoints for the service:

1. Metadata publishing endpoint.
2. Data Service access endpoint.

Address, binding and binding configuration are specified for Data Service access endpoint. There is a binding element in bindings section for the binding configuration named “DataService” used in Data Service access endpoint. For MetaDataExhange endpoint,

there is no binding configuration in bindings section. System defaults are used for the binding configuration used in this endpoint. The interface which service implements is given in contract attribute. BaseAddress element specifies the base addresses used by the service. The “address” attributes can be used as relative to the BaseAddress element like in this case.

<Behaviours> Element of Data Service configuration file

```
<behaviors>
    <serviceBehaviors>
      <behavior name="DataServiceBehavior">
        <serviceCredentials>
          <userNameAuthentication userNamePasswordValidationMode="Custom"
customUserNamePasswordValidatorType="WSSAF.Data.DataService+CustomUserName
Validator, WSSAF.DataService"/>
          <serviceCertificate storeLocation="LocalMachine"
                storeName="TrustedPeople"
                x509FindType="FindBySubjectDistinguishedName"
                findValue="CN=DS" />
        </serviceCredentials>
        <serviceMetadata httpGetEnabled="True"/>
        <serviceDebug includeExceptionDetailInFaults="True" />
      </behavior>
    </serviceBehaviors>
  </behaviors>
```

There is one behavior element in this section. “DataServiceBehaviour” is the behavior configuration of the Data service. Data Service supports custom Username authentication over WSSAF Store. Users are stored in WSSAF Store. “UserNamePasswordValidationMode” attribure is set to “custom” and class type names are given as “WSSAF.Data.DataService + CustomUserNameValidator with namespace. WCF runtime calls validate method of the specified class to authenticate the client with custom code. Authentication Service certificate are set with attributes like location and key value. “includeExceptionDetailInFaults” attribute in “Servicedebug” element specifies whether to include managed exception information in the detail of SOAP faults returned to the client for debugging purposes. The default is false. “httpGetEnabled” attribute in “ServiceMetaData” element specifies whether to publish service metadata for retrieval using an HTTP/Get request. The default is false.

<Bindings> Element of Data Service configuration file

```
<bindings>
    <wsHttpBinding>
      <binding name="DataService" openTimeout="00:10:00" sendTimeout="00:10:00"
        bypassProxyOnLocal="false" transactionFlow="false"
hostNameComparisonMode="StrongWildcard"
        maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
messageEncoding="Text"
        textEncoding="utf-8" useDefaultWebProxy="true" allowCookies="false">
        <reliableSession ordered="true" inactivityTimeout="00:10:00"
          enabled="false" />
        <security mode="Message">
          <message clientCredentialType="UserName" negotiateServiceCredential="true"
            algorithmSuite="Default" establishSecurityContext="true" />
        </security>
      </binding>
    </wsHttpBinding>
</bindings>
```

There is one binding element of system type WsHttpbinding. For WSHttpBinding, there are many attributes as shown here. All of them have system default values. We have included some of them here in this binding configuration. New values other than default values can be set in this section. More important settings are mode for security element which is set as Message and “clientcredentialtype” which is set as Username.

This binding configuration is used in the Data Service access endpoint.

4.5 HOSTING WSSAF SERVICES

WSSAF Services must be hosted within a run-time environment. This run-time environment activates the service and controls its context and lifetime. WSSAF Services can be executed in any Windows process that supports managed code that is written on .NET Framework.

There are many hosting options for the services. Developers choose the hosting environment that satisfies the service's deployment requirements. Some of the options we have tested are listed below;

1. Windows Console Application for test and development purposes only.

2. Internet Information Services (IIS) (Microsoft IIS) integrated with ASP.NET (Microsoft ASP.NET).

4.6 WSSAF DATA CONTRACTS

A data contract is an agreement between a service and a client that describes the data to be exchanged. The WSSAF Data Contracts are located in the WSSAF.Types project which is included in the WSSAF Base Libraries. Figure 4.3 illustrates all the data contracts used in WSSAF.

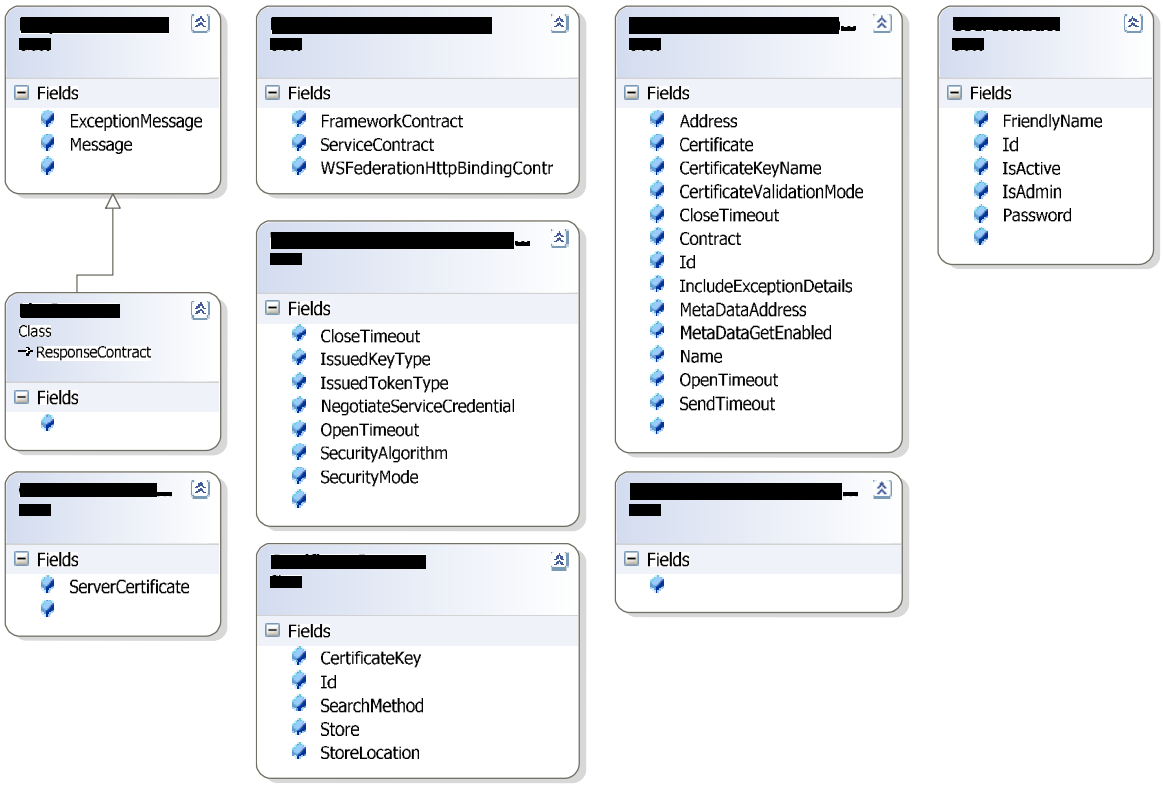

Figure 4.3. WSSAF Data Contract Classes.

There are 9 Data Contracts used in WSSAF:

(1) ResponseContract

Base contract class that can be used for all data contracts. It includes general properties that all contract needs. This is also used for a return type of a servis method execution.

(2) ListContract
It describes report data. (Result of query execution.)

(3) CertificatesContract
It describes the certificates raw data.

(4) RootConfigurationContract
It describes root contract definition for other contracts such as FrameworkConfiguration, ServiceConfiguration, WSFederationHttpBindingConfiguration.

(5) WSFederationHttpBindingConfigurationContract
It describes WSFederationHttpBinding Settings defined in WSSAF Store.

(6) CertificateContract
It describes certificate properties defined in WSSAF Store.

(7) ServiceConfigurationContract
It describes Service properties defined in WSSAF Store.

(8) FrameworkConfigurationContract
It describes root contract definition for WSSAF services con

(9) UserContract
It describes user properties defined in WSSAF Store.

4.7 WSSAF MANAGEMENT CONSOLE (WSSAF WEB)

The WSSAF Management Console is a web application to configure and manage information about users, services and certificates in the framework. To login into the WSSAF Web, a username and password is required. The password should be defined in the WSSAF store with an administrator role. Figure 4.4 illustrates the login page.

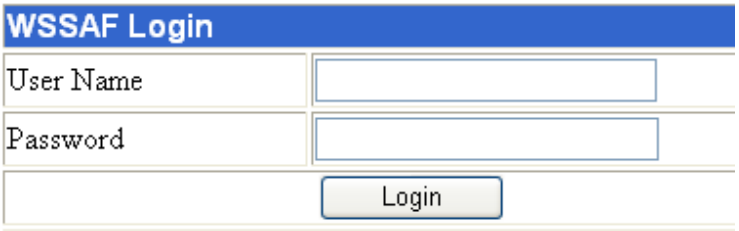

Figure 4.4. WSSAF Web login page.

After logging into the system, there are link buttons to manage users, service configurations and certificates. Figure 4.5 illustrates the user management page.

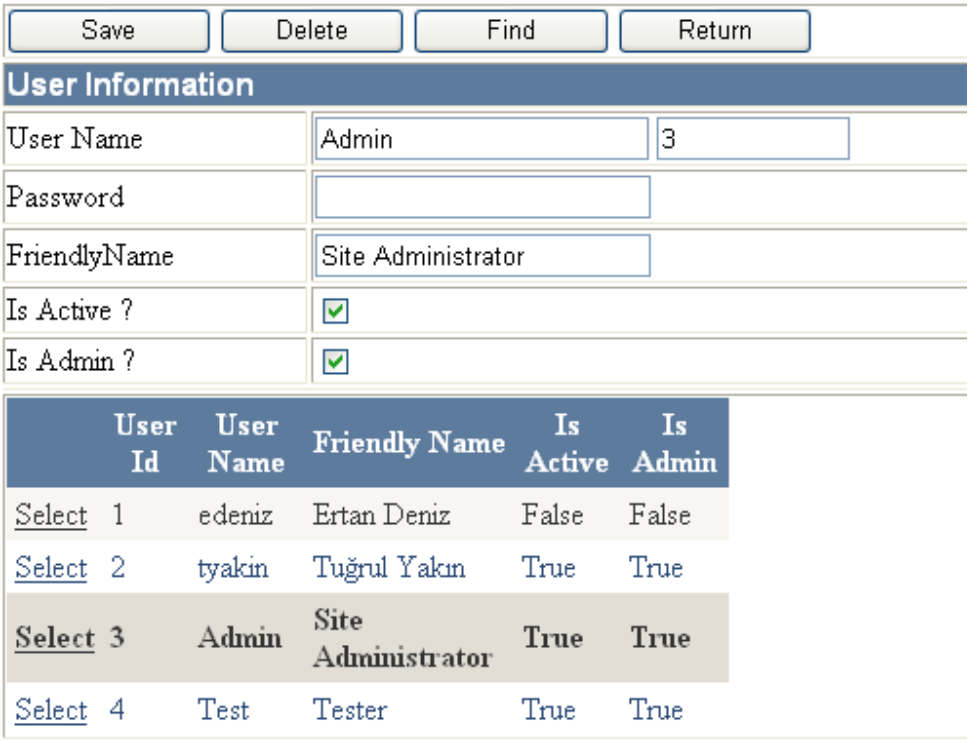

Figure 4.5. WSSAF Web Users Page.

Figure 4.6 illustrates the service configuration management page.

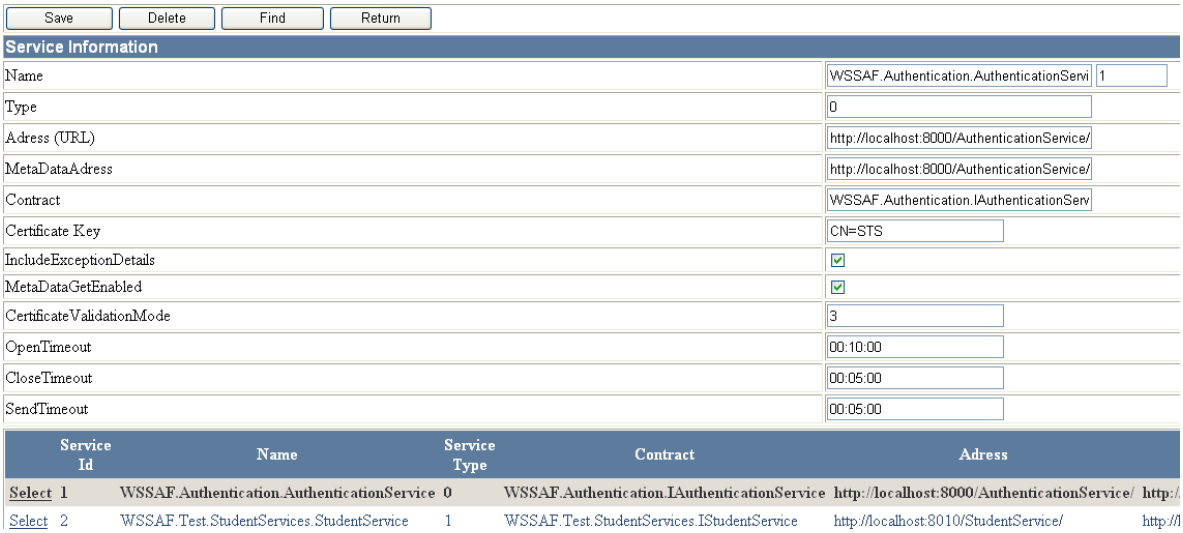

Figure 4.6. The WSSAF Web Service Configurations page.

WSSAF Web communicates directly with WSSAF Data Service to accomplish all the tasks assigned to it.

CHAPTER 5

TESTING WSSAF

A test service called Student Service were developed and defined in the WSSAF container using the WSSAF Web. Two test clients, one for Username and the other for Certificate authentication were developed to test the WSSAF Enabled Student Service. The development platform and the testing environment tools are the same as used in WSSAF.

In the following sub-sections, Test Service development and Test clients will be explained.

5.1 TEST SERVICE DEVELOPMENT

There are 6 steps to develop WSSAF enabled services;

(1) Develop the service with all the business functionality.

Test service has one method called "GetStudentById". The client calls GetStudentById method of the Student Service with a student id. This method returns student's name as a return value.

(2) Choose hosting options for Test Service.

The Student Service is hosted by a Windows Console Application. This option is selected for test and development purposes only.

(3) Configure the service to access WSSAF Data Service.

<Client> Element of Student Service configuration file

```
<client>
 <endpoint name="DataService" behaviorConfiguration="DataServiceBehavior"
        address="http://localhost:8020/DataService/"
        binding="wsHttpBinding"
        bindingConfiguration="DataService"
        contract="WSSAF.Types.IDataService">
  <identity>
   <dns value="DS" />
  </identity>
 </endpoint>
</client>
```

The Student service is a client for the WSSAF Data Service. An endpoint is defined to access the Data Service with address, binding, binding configuration and contract attributes. Behavior and binding configurations are included in the behaviors and bindings sections. The interface which the Data Service implements is given in the contract attribute.

The “Identity” element specifies settings that enable the client to authenticate the server when using an issued token. The identity element allows a client to specify the expected identity of the service. It includes DNS of Data Service X.509 certificate used to authenticate the service. In the handshake process between the client and service, the Windows Communication Foundation (WCF) will ensure that the identity of the expected service matches the values of this element.

<Behaviours> Element of Student Service configuration file

```
<behaviors>
   <endpointBehaviors>
    <behavior name="DataServiceBehavior"  >
     <clientCredentials supportInteractive="false" >
      <serviceCertificate>
       <authentication certificateValidationMode="PeerOrChainTrust"/>
        <defaultCertificate storeLocation="CurrentUser"
                                                    storeName="TrustedPeople"
                                        x509FindType="FindBySubjectDistinguishedName"
                        findValue="CN=DS"/>
      </serviceCertificate>
```

```
        </clientCredentials>
      </behavior>
    </endpointBehaviors>
  </behaviors>
```

Data Service certificate are set with attributes like location and key value.

<Bindings> Element of Student Service configuration file

```
<bindings>
    <wsHttpBinding>
      <binding name="DataService" openTimeout="00:10:00" sendTimeout="00:10:00"
allowCookies="false">
        <security mode="Message">
          <message clientCredentialType="UserName"/>
        </security>
      </binding>
    </wsHttpBinding>
  </bindings>
```

For WSHttpBinding, there are many attributes as can be seen above. All of them have default values. We have included some of them here, in this binding configuration. New values other than default values can be set in this section. Important settings are the "mode" in the security element which is set as Message and the "clientcredentialtype" which is set as Username.

(4) Add WSSAF Data Service access code in startup section of Windows console application to get service configurations dynamically.

The Data Services GetRootConfiguration method is called. This method returns RootConfiguration Contract with data loaded. This contract as the name implies describes the root contract definition for other contracts such as FrameworkConfiguration, ServiceConfiguration, WSFederationHttpBindingConfiguration. All the necessary configurations to host the service dynamically are loaded with a single request made to the WSSAF Data Service.

(5) Use WCF and WSSAF utilities to configure the service in code with dynamic configuration data.

First a service host environment class should be defined. WCF supports giving service configurations in code before service host environment is opened.

By using the ServiceHost class, endpoints, behaviors, bindings and other options are added or changed in code. Within this process, WSSAF utilities are used for preparing Authentication Service binding class and converting certificate raw data to X.509 certificate type.

(6) Open the service host environment.

Service host environment is opened that is open method is called.

5.2 TEST CLIENTS

Two test clients one for Username authentication and the other for Certificate authentication were developed to test the WSSAF Enabled Student Service. The clients get a security token from WSSAF Authentication Service and make synchronous requests to Student Service's IsStudentExist method with a student Id and Student Service replies with the result.

Configuration file for Test Client with UserName Authentication looks as follows;

<Client> Element of Test Client's configuration file

```
<client>
      <endpoint address="http://localhost:8010/StudentService/"
behaviorConfiguration="Client"
        binding="wsFederationHttpBinding" bindingConfiguration="Authentication"
        contract="WSSAF.Test.StudentServices.IStudentService" name="StudentService"
/>
   </client>
```

The test project is a client for the Student Service. An endpoint is defined to access to Student Service with address, binding, binding configuration and contract attributes.

Behavior and binding configurations are included in the behaviors and bindings sections. The interface which the Student Service implements is given in the contract attribute.

<Behaviours> Element of Test Client's configuration file

```
<behaviors>
    <endpointBehaviors>
      <behavior name="Client"  >
        <clientCredentials supportInteractive="false" >
          <serviceCertificate>
            <authentication certificateValidationMode="PeerOrChainTrust"/>
            <defaultCertificate storeLocation="CurrentUser"
                                storeName="TrustedPeople"
                    x509FindType="FindBySubjectDistinguishedName"
                           findValue="CN=localhost"/>
          </serviceCertificate>
        </clientCredentials>
      </behavior>
    </endpointBehaviors>
</behaviors>
```

There is one behaviour element called "Client". Student Service certificate are set with attributes like location and key value.

<Bindings> Element of Test Client's configuration file

```
<bindings>
      <wsFederationHttpBinding>
        <binding name="Authentication" openTimeout="00:10:00"
sendTimeout="00:10:00">
            <security mode="Message">
              <message issuedKeyType="SymmetricKey"
issuedTokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1">
          <issuer address="http://localhost:8000/AuthenticationService/"
                                                  binding="wsHttpBinding"
                                bindingConfiguration="stsWsHttpBinding">
                <identity>
                  <dns value="STS" />
                  <certificateReference storeName="TrustedPeople"
storeLocation="CurrentUser"
                   x509FindType="FindBySubjectName" findValue="CN=STS" />
                </identity>
              </issuer>
```

```
            </message>
          </security>
        </binding>
      </wsFederationHttpBinding>
      <wsHttpBinding>
        <binding name="stsWsHttpBinding">
          <security mode="Message">
            <message clientCredentialType="UserName"/>
          </security>
        </binding>
      </wsHttpBinding>
    </bindings>
```

There are two binding elements. One of them is of type WsFederationHttpBinding. WCF support for the WS-Federation protocol with this binding. This binding element contains binding attributes for accessing Authentication Service. Security mode, Address of Authentication Service, IssuedTokenType and IssuedKeyType are the important settings of this binding element.

Security mode is set to “Message” for message security.

IssuedTokenType is set to URI value “http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1” that specifies the type of token.

IssuedKeyType is set to “SymmetricKey” value.

The other binding element is of type WsHttpbinding. This binding element contains binding attributes for accessing Student Service. More important settings are “mode” for security element which is set as Message and the clientcredentialtype which is set as Username.

Configuration file for Test Client with Certificate Authentication is nearly same as the above configuration. There is one difference in the WsHttpBinding elements bindings section. The clientCredentialType is set to “Certificate” value. The change is shown with the binding element below;

```
<wsHttpBinding>
      <binding name="stsWsHttpBinding">
        <security mode="Message">
```

```
        <message clientCredentialType="Certificate"/>
      </security>
    </binding>
</wsHttpBinding>
```

One more point worth mentioning is that our client certificate has the key value set as “Client.com” within the code; we could set it in the configuration file as well.

CHAPTER 6

RELATED WORK

Some of the researchers have developed API based toolkits. A Web Services Security Framework (WSSF) is presented by (Peng and Wu, 2006). This framework offers a solution for secure communication and access control of Web services in the defined Web Services container. Some of the main features presented in this work are the SOAP message processing layer at both the server side and the client side, authorization and permission delegation. An application programming model WSSAPI (Yamaguchi et al, 2007) was proposed to provide easy set up of WS-Security configurations. It looks at the WS-Security processing from an abstract level. The comparison of WSSAPI to other toolkits is discussed. The Web Services Security toolkit (WCF) that we have used provides most of these features out of box and gives a good level of abstraction to hide the complexity of Web Service Security configuration according to WS-* specifications.

Some of the researchers focused on developing Secure Web Services. A Single Sign-On (SSO) web service as a solution to existing problems with web based authorization and authentication was proposed by Challagulla (Challagulla, 2004). A "Data Operation Web Service" using the component based development architecture was developed by Park et al. (Park et al, 2007) Their objective is to improve the web service security by adding security features and functionalities such as Auditing, Certification and Authentication.

Besides the work mentioned above, tools and frameworks for generating configurations to provide Web Service security were developed. A tool for generating WS-SecurityPolicy descriptions that express security policies and requirements for a service was developed by Tatsubori et al. (Tatsubori et al, 2004) A tooling framework to generate

Web services security configurations using Model Driven Architecture (MDA) for making configuration management simple and increasing the usability was propsed by Nakamura et al. (Nakamura et al, 2005) Security requirements are added to the application model, and then detailed security configurations are generated. A Model-Driven Security framework that generates platform specific configuration files for WS-Security by transforming a security policy was proposed by Satoh and Yamaguchi. (Satoh and Yamaguchi, 2007) They proposed to use a generic security policy transformation framework for platform-specific configurations via intermediate models.

Agent based security policy frameworks were developed as well. They provide policy languages such as ReiT. ReiT is a declarative language based on the rules and ontologies introduced by (Li et al, 2007). Li et al proposes a mixed reasoning mechanism to evaluate the ReiT policy. The access control policy including the context of the user and Web Services is evaluated by the reasoner. In addition, they present a policy aware agent to authorize the access control of the Web Services. A core ontology and a logic-based situation-aware security specification language for specifying dynamic security policies for service-based systems are provided by (Yau et al, 2005). With their approach, they detect the policy conflicts and resolution. They also provide tools for generating and deploying security agents to enforce security policies.

In the literature we haven't recognized an end to end Security Framework Service including Authentication, Authorization, Auditing, Exception Management and Reporting by utilizing the power of Web Services. In our framework, we have broadly modeled the required functionality as Security Web services for securing the Web services in the container. We have utilized the Web services development toolkit (WCF) which provides standards based functionality. We have preferred to provide the Web Service security configurations dynamically during the start up phase of the service. The Web Service configurations are managed using the Web Management Console. We aimed to provide the advantages of being lightweight, distributable and an easy configurable SOA security framework to encourage the business domains to make an easy transition to the Web Services world.

CHAPTER 7

CONCLUSIONS

In our work we have proposed a Service Framework Model and developed the Authentication and Data services. We have implemented test services and clients to show the feasibility of the framework. Our further work will include the development of all the Security Services with acceptable performance. Our next concern will be interoperability. Web services developed in other platforms with different toolkits will be able to utilize our framework.

After all Security Framework Services have been developed, WSSAF can be easily used to protect Web Service resources in business domains like universities, hospitals and e-Government agencies.

With this study, we present Web Services Based Security Application Framework (WSSAF) for securing Web services. In SOA environments, there are many security configurations for different security requirements. In addition, Web Services Security standards are growing rapidly. There is urgent need to abstract the complexity and increase usability of WS-* standards. WSSAF provides a lightweight, distributable and easy configurable framework by using the power of Web Services. The WSSAF model encourages the companies to take big steps into Web Services world with no worry about security.

The major contributions of this study are (1) To encourage to use Web services and Web Services Security existing standards. (2) To show that Web Services Security is the first order business consideration to get into the Web Services world. (3) To package Security functions as Web Services in a Framework with main features like being

lightweight, distributable and easy configurable by utilizing the power of Web Services. (4) To provide reusability at service level. (5) To make security ready at the beginning of any SOA Project. (6) To assist on providing organizational standards by sharing the security services for different projects in the same domain.

Business Domains like universities and hospitals can utilize WSSAF to add Web Services Security based on current standards to their Web Services Investments. We have shown that WSSAF based on the Service Oriented Modeling approach is flexible enough over application Frameworks based on traditional class libraries approach. By utilizing the Web Services development toolkit (WCF), we have been comfortable during the development and testing phases.

REFERENCES

Adams C.,Farrell S., *Internet X.509 Public Key Infrastructure Certificate Management Protocols,1999,* http://www.ietf.org/rfc/rfc2510.txt

Bartel M., Boyer J., Fox B., LaMacchia B., Simon E., *XML Signature Syntax and Processing*, 2008,http://www.w3.org/TR/xmldsig-core/.

BEA Systems, BMC Software, CA, Inc., IBM, Layer 7 Technologies, Microsoft, Novell, VeriSign, *Web Services Federation Language (WS-Federation)*, 2006, http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-Federation-V1-1B.pdf?S_TACT=105AGX04&S_CMP=LP,

Booth D. et al, *W3C Web Services Architecture*, 2004, http://www.w3.org/TR/ws-arch/

Box D. et al, *Web Services Policy Assertions Language*, 2003, http://xml.coverpages.org/ws-policyassertionsV11.pdf

Bray T. et al. *Extensible Markup Language (XML). W3C Recommendation*, 2008, http://www.w3.org/TR/xml/

Brekken L. A., Åsprang R. F., *Adding Security to Web Services - An Automatic, Verifiable, and Centralized Mechanism for Web Services Input Validation*. Master of Science in Communication Technology, Norwegian University of Science and Technology, 2006

Cantor S. et al, *Security Assertion Markup Language (SAML) Version 2.0*, 2005, http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

Challagulla S., *Design and Development of an Authorization and Authentication Web Service*, Master of Science in Technology, Division of Computing Studies, Arizona State University.2004.

Chen X,*Developing Application Frameworks in .NET*. Apress,2004.

Chinnici R., Moreau J.J., Ryman A., Weerawarana S., *Web Services Description Language (WSDL) Version 2.0 Part1:Core Language*, 2007, http://www.w3.org/TR/wsdl20/.

Chumbley R., Martin, M.J., WS-I, Basic Profile 1.2 Interoperability Scenarios, Working Group Draft http://www.ws-i.org/docs/BP12_Interop_Scenarios.pdf, 2008.

Clement L., Hately A., Riegen C. V., Rogers T., *UDDI Version 3.0.2*, 2004, http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf.
Ford W., Baker P. H., Fox B., Dillaway B., LaMacchia B., Epstein J., Lapp J., *XML Key Management Specification (XKMS)*, 2001,http://www.w3.org/TR/xkms.

Fremantle P., Patil S., *Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.1*, 2007, http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.1-spec-os-01.pdf.

Gudgin M., Hadley M., Mendelsohn N., Moreau J.J, Nielsen H. F., Karmarkar A., Lafon Y.**,** "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)", http://www.w3.org/TR/soap12-part1/ , 2007.

Hirsch F., *Getting Started with XML Security,2002,* *http://www.sitepoint.com/article/getting-started-xml-security/*

IBM , Microsoft, *Security in a Web Services World: A Proposed Architecture and Roadmap*, 2002, http://download.boulder.ibm.com/ibmdl/pub/software/dw/library/ws-secmap.pdf.

Imamura T., Dillaway B., Simon E., *XML Encryption Syntax and Processing*, 2002, http://www.w3.org/TR/xmlenc-core.

Lawrence K., Kaler C.,*WS-Security Core Specification 1.1*, 2006, http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

Lawrence K, Kaler C., *Web Services Security UsernameToken Profile 1.1*, 2006, http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-UsernameTokenProfile.pdf

Lawrence K, Kaler C., *Web Services Security X.509 Certificate Token Profile 1.1*,2006, http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-x509TokenProfile.pdf

Lawrence K, Kaler C., *Web Services Security Kerberos Token Profile 1.1*,2006, http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf

Lawrence K, Kaler C., *Web Services Security SAML Token Profile 1.1*,2006, http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf

Lawrence K, Kaler C., *WS-Trust 1.3 Specification*, 2007, http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf.

Lawrence K, Kaler C., *WS-SecureConversation v1.3 Specification*, 2007, http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf.

Lawrence K, Kaler C., *WS-SecurityPolicy 1.2*, 2007, http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.html.

Li J.X., Li B., Li L., Che T.S. "An Agent-based Policy Aware Framework for Web Services Security", *IFIP International Conference on Network and Paralel Computing – Workshops*,2007

Löwy J.,*Programming WCF Services 2nd Edition*. O'REILLY, 2008

Meier J.D., Farre C., Taylor J., Bansode P., Gregersen S., Sundararajan M., Boucher R., *Improving Web Services Security*, http://www.codeplex.com/WCFSecurityGuide.

Microsoft Patterns & Practices Group, *Scenarios, Patterns, and Implementation Guidance for Web Services Enhancements (WSE) 3.0*, 2005, http://msdn.microsoft.com/en-us/library/aa480545.aspx.

Microsoft Technology, *Windows Platform*, http://www.microsoft.com/windows/

Microsoft Technology, *Windows Communication Foundation*, http://msdn.microsoft.com/en-us/library/ms735119.aspx

Microsoft Technology, *.NET Framework 3.5*, http://msdn.microsoft.com/en-us/library/w0x726c2.aspx

Microsoft Technology, *Visual Studio 2008*, http://msdn.microsoft.com/en-us/library/aa187919.aspx

Microsoft Technology, *C# programming Language*, http://msdn.microsoft.com/en-us/library/kx37x362.aspx

Microsoft Technology, *ASP.NET*, http://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx

Microsoft Technology, *Internet Information Services(IIS)*, http://msdn.microsoft.com/en-us/library/aa902517.aspx

Microsoft Technology, SQL Server 2005, 2005, http://www.microsoft.com/sqlserver/2005/en/us/default.aspx

Moses T., OASIS Standard - *XACML 2.0 Specification Set*, 2005, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

Nakamura Y., Tatsubori M., Imamura T., Ono K.,"Model-Driven Security Based on a Web Services Security Architecture", *Proceedings of the 2005 IEEE International Conference on Service Computing (SCC'05)*.

Nakayama K., Ishizaki T., Oba M., "Application of Web Services Security Using Travel Industry Model", *Proceedings of the 2005 Symposium on Applications and the Internet Workshops (SAINT-W'05)*

Newcomer E., Robinson I., *Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.1*, 2007, http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-os.pdf.

OASIS Community. http://www.oasis-open.org/home/index.php

Papazoglou M. P, "Service -Oriented Computing: Concepts, Characteristics and Directions", *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03)*, IEEE 2003.

Papazoglou, M.P., "Web Services Technologies and Standards", ACM Computing Surveys, 2006

Papazoglou M. P., Traverso P., S. Dustdar, F. Leymann, "Service Oriented Computing : State of the art and research challenges",*IEEE Computer Society*, 2007.

Park E.J., Kim H.K., Lee R. Y. "Web Service Security model Using CBD Architecture", *Fifth International Conference on Software Engineering Research, Management and Applications*, IEEE 2007.

Peng Y., Wu Q., "Secure Communication and Access Control for Web Services Container", *Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06)*, 2006.

Resnick S., Crane R., Bowen C., *Essential Windows Communication for .NET Framework 3.5*, Addison-Wesley, 2008.

Satoh F., Yamaguchi Y., "Generic Security Policy Transformation Framework for WS-Security", *2007 IEEE International Conference on Web services (ICWS 2007).*

Schepers T., *A view on web service security : Can current security standards adequately secure web services?*, Final Thesis, University of Tilburg.

Sprott D.,Wilkess R.,Veryard R.Stephenson J.,*A CBDI Report series – Guiding the transition to web services and SOA*, 2003, http://www.cbdiforum.com/bronze/downloads/ws_roadmap_guide.pdf

Sun L., Li Y., "XML and Web Services Security", 2008 IEEE

Tatsubori M., Imamura T., Nakamura Y., "Best-Practice Patterns and Tool Support for Configuring Secure Web Services Messaging", *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*.

Vedamuthu A. S., Orchard D., Hirsch F., Hondo M., Yendluri P., Boubez T., Yalçinalp Ü., *Web Services Policy 1.5 - Framework*, 2007,http://www.w3.org/TR/ws-policy/.

Vedamuthu A. S., Orchard D., Hirsch F., Hondo M., Yendluri P., Boubez T., Yalçinalp Ü., *Web Services Policy 1.5 - Attachment*, 2007, http://www.w3.org/TR/ws-policy-attach/.

W3C Specification, HTTP - *Hypertext Transfer Protocol*, 2009, http://www.w3.org/Protocols/

Yamaguchi Y., Chung H.V., Teraguchi M., Uramoto N., " Easy-To-Use Programming Model for Web Services Security", *IEEE Asia-Pacific Services Computing Conference*,2007

Yau S. S., Yao Y., Chen Z., Zhu L., "An Adaptable Security Framework for Service-based Systems", *Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'05)*

You X., Chen X., Fang X., Dang X., Zhou B., Wei B., "Web Services Security in Data Service Delivery Platform", *Proceedings of the IEEE International Conference on E-Commerce Technology for Dynamic E-Business (CEC,East'04)*

APPENDIX A

WEB SERVICES SECURITY GLOSSARY

Term	Definition
Assertion	Assertion contains a packet of security information including authentication, attribute and authorization information related with the subject.
Authentication	Authentication is the process of verifying identification credentials supplied by a user and validating these credentials from a authority. Authentication provides that calling a Web service is allowed only for authenticated clients.
Auditing	Auditing is the process of recording security-related events.
Claim	A claim is a declaration made by an entity that shows the capabilities of the entity.
Client	A user that accesses the Web service.
Credential	Credential is used to manage access for information resources. Username is a credential.
Digital signature	A digital signature is an electronic signature that can be used to authenticate the client that sends the message and to ensure that the message is unchanged.

Message	A message is a basic unit of communication that may consist of several parts, including a body and headers.
Message Encryption	Message encryption is the process to protect a message by converting the contents to cipher text using cryptographic methods.
Message integrity	Message integrity is the process of guaranteeing a message remain unaltered during transmission.
Message confidentiality	Message Confidentiality is the process of ensuring that message is confidental and it can not be monitored by unauthorized clients or anyone that monitors the information flow in the network. Encryption is the method used to enforce confidentiality.
Message layer security	Message layer security is the way where all the security information is transmitted in the message.
Message Signing	Message signing is the process of signing a message with a digital signature using cryptographic methods to confirm the source of the message and control if the message content have been tampered in authentication process.
Non-repudiation	Non-repudiation is the assurance that a user cannot deny performing an operation or sending a message. Detailed auditing and logging is the main functionality that every system must implement to provide non-repudiation.
Security Token	Security token is a unit of information that represents security-related information like X.509 certificate, Kerberos tickets, username.

Security token service (STS)	A Web service that issues security tokens.
Service	A service is a software system that exposes methods and support interaction with clients.
SAML	Security Assertion Markup Language (SAML) is an XML-based standard for exchanging authentication and authorization information between security domains, which includes an identity provider and a service provider.
SOAP	Simple Object Access Protocol is a specification for exchanging messages in the Web Services platform.
X.509	X.509 is published as ITU recommendation which defines a standard certificate format for public key infrastructure.
XML Encryption	The XML Encryption is a specification that defines how to encrypt the contents of an XML element. The specification contains a standard model for encrypting both binary and textual data.
XML Signature	A specification that defines a XML syntax for digital signatures. Digital signatures become a persistent part of the document. It also provides a means of communicating that the message contents were not altered during transmission.
XKMS	XML Key Management Specification (XKMS) is a protocol which defines XML message formats for public key management.
XACML	XML Access Control Markup Language (XACML)

	defines an XML vocabulary to specify the rules on which access control decisions are made.