DEGREE PROJECT 2004:PM01

Division of Computer Science at the Department of Informatics and Mathematics

Securing XML Web Services
-using WS-Security

Martin Antonsson

HÖGSKOLAN
TROLLHÄTTAN • UDDEVALLA

DEGREE PROJECT

University of Trollhättan · Uddevalla
Department of Informatics and Mathematics

Degree project for master degree in Software engineering

Securing XML Web Services

-using WS-Security

Martin Antonsson

Examiner:
Stefan Mankefors Department of Informatics and Mathematics

Supervisor:
Stefan Mankefors Department of Informatics and Mathematics

Trollhättan, 2003

2004:PM01

EXAMENSARBETE

Securing XML Web Services

-using WS-Security

Martin Antonsson

Sammanfattning

Teknologin XML Web Services (Webbtjänster) bygger på att data skickas via ett nätverk, expempelvis via Internet. Detta sker oftast genom brandväggar. Det är i kommunikationen mellan de inblandade parterna som problem uppstår. HTTP i kombination med SSL till exempel, kan endast garantera säkerheten från en punkt till en annan. Men när XML Web Services kommer in i bilden är kommunikation något mer komplex. Data är inbäddad i ett SOAP meddelande och detta kan skickas med flera transport protokoll, inte bara HTTP. Ett exempel på ett sådant är SMTP. Meddelandet kan också färdas mellan flera mellanhänder och denna topologi kräver ett sätt att säkra kommunikationen hela vägen, från start till slut.

Om ett företag applicerar en implementation av WS-Security skulle kommunikationen bli säker från start till slut. WS-Security är en flexibel specifikation. Den specificerar att ett företag ska använda sig utav existerande standarder och tekniker för att säkra meddelandets konfidentialitet och integritet. För att uppnå detta rekommenderar WS-Security att använda XML Encryption respektive XML Signature. WS-Security stödjer också flera sk. security tokens som används för att autenticera slutanvändaren.

Denna magisteruppsats beskriver en implementation av specifikationen WS-Security. Implementation är genomförd hos Fishbone Systems AB. Om de applicerar projektets implementation, kommer deras XML Web Services att förbli säkra.

Utgivare:	Högskolan Trollhättan · Uddevalla, Institutionen för Informatik och Mattematik Box 957, 461 29 Trollhättan Tel: 0520-47 50 00 Fax: 0520-47 50 99		
Examinator:	Stefan Mankefors		
Handledare:	Stefan Mankefors, HTU & Jörgen Bjerkesjö, Fishbone Systems AB		
Huvudämne:	Programvaruteknik	**Språk**:	Engelska
Nivå:	Fördjupningsnivå 2	**Poäng**:	10
Rapportnr:	2004:PM01	**Datum**:	2003-12-19
Nyckelord:	Security, XML Web Service, WS-Security, XML Encryption		

Innehållsförteckning

Securing XML Web Services
-using WS-Security

Martin Antonsson
University of Trollhättan/Uddevalla
ds99maan@thn.htu.se

Abstract

The technology XML Web Services builds upon that data is transferred on a network like the Internet. It usually occurs behind firewalls. It is in the communication between parties involved where problem arise. HTTP in combination with SSL for instance can only guarantee the security from one point to another, but when XML Web Services come into the picture the communication is often more complex. The data is embedded in a SOAP message and it can be sent over several transport protocols, not only HTTP. SMTP is one of them. The data may also travel between multiple intermediaries and this topology demands a way to secure the complete communication, end-to-end.

If a company adopts an implementation of WS-Security the security in the communication could be guaranteed end-to-end. WS-Security is a specification designed to be flexible. It specifies using existing standards and techniques for securing the message's confidentiality and integrity. For this, WS-Security endorses to use XML Encryption and XML Signature respectively. It also supports multiple security tokens that can be used to authenticate the end user.

This master thesis describes one implementation of WS-Security. The implementation is conducted at Fishbone Systems AB. If they apply this project's implementation, their XML Web Services will be secured.

1. Introduction

Fishbone Systems AB is an IT-company that offers solutions for process automation to clients striving for performance improvement through process orientation. The clients' main processes are supported by IT systems designed for the purpose, and processes can be altered with little or no changes in IT systems. Fishbone Systems AB uses XML Web Services in their software development which facilitates development of a solution where a client's applications can be integrated with each other.[1]

Fishbone Systems AB develops mostly web applications, which are accessible via an Intranet and/or the Internet. It is in the communication between parties involved where problem arises. The technology behind XML Web Services builds upon that data is sent via a network, like the Internet, between a sender and a receiver. The most common protocol for sending data via the Internet is the Hyper Text Transport Protocol (HTTP). HTTP, combined with for example SSL, only delivers security when data is sent from one point to another; in more complex environments where XML Web Services have to collaborate with several intermediaries before the message reaches the final destination, there has to be a way to guarantee the security. The complete transmission of data, end-to-end, has to be secure.[2]

The specification WS-Security specifies directions to use existing standards for achieving secure XML Web Services. Securing the confidentiality and the integrity of a message using XML-Encryption and XML Signature respectively are endorsed by WS-Security.[3]

1.1 Objectives

The general objective of this master degree project is to:

- Investigate the security support WS-Security brings to a company when developing secure XML Web Services.

- Show how a company like Fishbone Systems AB can build secure XML Web Services according to WS-Security. For that reason, the degree project also includes a practical implementation where an XML Web Service can communicate in a secure way with an application. This was conducted at Fishbone Systems AB.

1.2 Disposition

The degree project is divided in two parts:

Part 1
To clarify what security supports WS-Security enables for a company, the specification is described in part 1. This underlies how one applies WS-Security when implementing secure XML Web

Services. The security regarding XML Web Services and the threats the technology is exposed to is described in part 1 as well.

Further, the technology XML Web Services and the main components behind it are described in part 1. These are Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI).

Part 2
The practical implementation is described in part 2. A fictive credit account service was created according to WS-Security. It illustrates how to solve those problems that a company like Fishbone Systems AB stands before when adopting XML Web Services in their development.

Bearing in mind that there is no standard to follow when building secure XML Web Services the degree project only comes up with a suggestion of an approach.[4] The implementation is suitable for a company like Fishbone Systems AB.

1.3 Restrictions

Like in the real world when a money transport is on its way from a company to a bank, the vehicle and the guards in the vehicle are responsible for the security. They are not using a secure pipeline between the company and the bank where money could be sent via. It should be the same scenario in the virtual world; the transport protocol should not be responsible for the security when a message is transferred.

Instead, the degree project focuses on applying the security on the SOAP message itself, not on the transport protocol on which the SOAP message is sent. In those cases when an application makes a request to an XML Web Service, the result is embedded in a SOAP message and is sent back to the calling application. This could be inside an Intranet or on the open Internet. If one then places the responsibility for handling the security on the SOAP message, the message can be sent via different transport protocols. To achieve secure communication with the help of a SOAP message, the SOAP message itself must be manipulated in some how. This could be that the content of the body in the SOAP message is encrypted. This ensures that unauthorized eyes will have trouble viewing the content. For increasing the security further, the header of the SOAP message could contain a user name and password, which could verify if the client application has the permission to access the XML Web Service.[5]

The practical implementation has been performed in the .NET environment. The result are based on a Microsoft XML Web Service and an ASP .NET application developed in Visual Studio .NET. The degree project will not consider development of XML Web Services with Java in the J2EE environment.

1.4 Background

A white paper written by IBM together with Microsoft describes that the security around XML Web Services has to be improved. IBM and Microsoft expect to work closely with customers, partners and the industry for improving the security model, Web Service Security (WS-Security).[6]

There are different alternatives to create a secure solution with XML Web Services. One method for a company, which this degree project does not cover in detail, is to ensure that the connection between the XML Web Services, intermediaries and applications are secure. This can be done by a selection of techniques and three of these ways are; use restriction rules in the firewalls, use Secure Sockets Layer (SSL) or use a Virtual Private Network (VPN).[7]

If a company knows which computers in a network that have the authorization to access the XML Web Services, the configuration in the firewalls can be set to allow the IP addresses from the specific machines. This technique can be useful in a private network but if the company works globally; it is nearly impossible to restrict IP addresses because it is not sure that the clients have an intact IP address. One can also use a firewall such as Microsoft Internet Security and Acceleration (ISA) to give specific clients permission to access certain methods or functionality in the XML Web Services.[7]

The second alternative is to use SSL to establish a secure connection on the network. SSL encrypts the message from the sender and when it reaches the receiver, SSL decrypts the message. In this way one ensures that the message was not read while transferred.[7] But SSL combined with HTTP only provides secure transmission between two points and when working with XML Web Service one has to find a way to secure the transmission end-to-end.[5]

The third alternative that were listed earlier and that this degree project does not consider is to use VPN for securing the connection. Also VPN enables a secure connection, like a tunnel, between two computers on a network. In this way the computers that are connected to each other via a VPN can securely communicate.[7]

When an end user, that is, a human behind the computer, makes a request to a portal on the Internet, the user does not make the actual SOAP message. However, if the Web Service that the user indirect tries to communicate with need specific information about the end user for deciding whether the user has the permission to access the Web Service or not, the Web Service must have information about the end user. The portal handles the communication with the Web Service and not the end user. For example if a user enters the portal and makes a reservation of some sort. The reservation is made, on the user's behalf; via the portal that makes the SOAP request. The end user may have been authenticated to the portal, maybe by entering a user name and password in a form. The portal knows information about the user's identity, as well as attributes of the end user such as previous reservations. However, in this case the Web Service has only visibility of the portal, not the end user. How can the information about the end user be passed on to the Web Service? Neither session layer nor transport layer applies security between the portal and the Web Service regarding information about the identity of the end user of the Web Service. It merely expresses information about the portal that is sending the SOAP message. It may be the case that many of the requests to the Web Service originate from the portal. This challenge is all about placing the responsibility of the security on the SOAP message. The information that must be presented for the Web Service may concern the end user's identity, maybe a user name and password or simple an indication that the end user is authenticated and/or authorized by the portal. It is such information, which the Web Service needs to receive to make a decision if the end user has the permission to access the Web Service. This scenario is likely to be widespread where many Web Services are used to perform tasks for businesses, in this case the portal. It should not be the case that the end user has to re-authenticate each time a SOAP request are sent on their behalf. The challenge of providing this functionality is sometimes called single-sign-on or federated trust. An example of this is Microsoft .Net's Passport. One user can sing in on one place and move to another web site without have to sign in at the other place.[8]

2. Part 1

2.1 XML Web Services

Like every new technology that arises inside the IT-industry, XML Web Services stands before widespread testing and evaluating from developers, vendors and customers. The scepticism for a new technology has always been high. Now everybody waits and sees if the technology XML Web Services is here to stay. The terms Web Services and XML are usually put together and one can get the impression that Web Services are built upon XML. In fact, without the Extensible Markup Language (XML) we would not have seen the XML Web Services in action. The cornerstones of XML Web Services are all built upon XML. That is why XML is a big part of this technology. The components are Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI). They will be described in more detail in the following sections.[9]

There are different opinions and definitions of an XML Web Service and one that the author Ethan Cerami defines is:

"A Web Service is any service that is available over the Internet, uses a standardized XML messaging system, and is not tied to any one operating system or programming language."[10]

The idea behind XML Web Services is to make two independent systems to interact and integrate with each other. This is not a new and spectacular invention because Common Object Request Broker Architecture (CORBA) and the distributed extension of Microsoft's COM (Component Object Model), DCOM, which are similar technologies, have been on the market for some years. CORBA is the Object Management Group's specification for achieving interoperability between two distributed systems. These systems could be placed in heterogeneous environments and communicate at the object level. DCOM on the other hand, builds an object remote procedure call to support remote objects. A COM client can interact with a COM object via its methods and interfaces and invoke these methods for performing different tasks. In this way, it looks like the objects reside on the same place as the client but in fact; the applications can be placed on different places, for example in a client/server solution. Given that, the design is at the binary level, it makes it possible for components written in different programming languages such as C++, Java, and Visual Basic for example to communicate and share data.[11] Sun's alternative for communication between two objects is Remote Method Invocations (RMI). RMI is a mechanism that allows components to invoke methods on objects that resides on a different storage place. However, RMI is only useful for communication between two Java objects; the client machine and the server machine must have a Java Virtual machine running.[12]

Many had the expectation that companies would choose either one of the technologies CORBA and

DCOM as a universal standard for interoperability between two heterogeneous systems. But migration to a different platform costs industry time and money and they had already made big investments in platforms supporting specific distributed-computing technologies. CORBA has also had a reputation for slow performance and complex implementations. Microsoft decided from earlier experience with interoperability to base the Web Services on open standards that could communicate with any platform. That is why Microsoft chose XML and other well-known standards.[9][12]

The structure of an XML Web Service can be described in different ways. A demand is that the application shall be accessible over the Internet so other XML Web Services can exchange data with it. An XML Web Service is usually divided in 4 main layers. These are:[10]

Service transport
This layer is responsible for the communication between the applications. It uses the protocol HTTP for sending messages via the Internet and enables two XML Web Services distributed on the Internet to exchange data. [10]

XML Messaging
This layer makes it possible for the messages that are sent via the transport layer to be viewable for both receiver and sender. The messages are encoded in a common XML format that both ends can understand. This layer also includes the XML-based protocol SOAP. [10]

Service description
Every XML Web Service is self-described. The interfaces of the XML Web Service are described in a language that is called Web Service Description Language (WSDL). [10]

Service discovery
The XML Web Service is provided by a registry. This layer is responsible for placing the XML Web Service in this registry and it also supplies publish and find functionality for the specific XML Web Service. For the moment, this service is handled via Universal Description, Discovery and Intergation (UDDI).[10]

2.2 XML – Extensible Markup Language

Understanding XML is essential to obtain an insight in how XML Web Services work. It provides the XML Web Services a platform independent, flexible and extensible mark up technology.[9]

XML is a standard developed by the World Wide Web Consortium (W3C) to create a universal format for handling data and exchanging data between applications. Additionally, one uses XML to store data in a structured way. The applications can be positioned on different platforms as long as they recognise the XML format. This makes the XML format platform independent. In 1998, the XML version 1.0 was accepted and from that point, the technology was stable for deployment in industry. [9]

XML is used to build a document and structure the information with a clear marking. XML is an extension of the Standardized General Markup Language, SGML and is also closely related to the Hyper Text Markup Language, HTML. It combines the power and extensibility of its related languages. The structure of an XML document is constructed of the tags that describe the data, attributes and comments. Elements are used for marking up sections in the document and the data that the user wants to structure are written between the tags. Attributes describe units of the content and the comments are there for make the code clear but ignores from the application.[9]

If one wants to store information about persons and describe a person with the attributes age, weight, height; an XML document could look like this:[13]

```
<?xml version=”1.0”>
<persons>
  <person>
     <age>24</age>
       <weight
       measureWeight=”kg”>70</weight>
       <height
       measureHeight=”cm”>180</height>
  </person>
</persons>
```

It is a tree structure in the XML document, where the elements are nested inside of other elements in order to form a hierarchy. The one element at the top is called the root element and all the others are called subelements of the root. To describe complex data, the children have subelemets of their own. There is no limit to the complexity and total of subelements.[14] In the example above, is it easy to add another person and illustrate the person’s details. The attributes `measureWeight` and `measureHeight` indicates kilogram and centimetres respectively. If the local authority stores their residents in the same way, it is easy to share data amongst each other. If two authorities adopt the same syntax using XML, a web site that handles the search of the residents in both districts could be easily built.

There can only be one root element, all other elements must be correctly nested and one attribute can only have one value. To ensure that the document is according to the vocabulary’s rules one connects the XML document to a template that describes the rules. Two templates that record the

rules are DTD and XML Schema. The XML Schema is flexible to use and easy to grasp because they use XML syntax. One simple refers to the specific XML Schema that the XML document shall follow inside the XML document.[14]

2.3 SOAP – Simple Object Access Protocol

SOAP is a protocol and as its name also reveals it focuses on objects. It enables communication via objects distributed across a network, for example over the Internet. That object can be anything from an application, such as Microsoft Excel, to a Web Service. The report mention before that SOAP was based on XML. [14] Although SOAP can be used in many different messaging systems and be delivered via the protocol Simple Mail Transport Protocol (SMTP), the focus is to use HTTP as the transporting protocol. This is just because XML Web Services need a way to operate and get through the firewalls. The HTTP usually works on port 80 and this one is open default in some of the firewalls' configuration. Accordingly, SOAP uses XML to mark up data and HTTP for transporting data which makes it platform independent. [9]

SOAP specifies a messaging framework like a software system where applications can communicate with each other by exchanging messages. They send XML documents called SOAP messages over the network. The messages contain information that is transmitted to and from a Web Service. The messages do not include programming instructions; instead they tell the specific Web Service which method to invoke. In this way the desired tasks are performed.[9]

SOAP works after a one-way message model, that is, the messages represent either a SOAP sender or a SOAP receiver. In the SOAP model, components on the network that understand SOAP, which are called nodes, process SOAP messages. The messages are transported on a message path, where several nodes can work as intermediaries.[15]

SOAP consists of three main parts:[15]

-**The SOAP envelope** defines the structure of a SOAP message, a specific type of XML document. [15] The SOAP envelope is described in more detail in Appendix A.

-**SOAP specifies encoding rules**. These rules describe how data can be represented in a SOAP message. The rules help the receiving application to recognize the format of the SOAP message and in this way the application can process the data in the message properly. Developers can use any encoding method, as long as they point out the rules they are working with. This is because SOAP does not force developers to use a particular programming language and different programming languages interpret different encoding rules. The flexibility within SOAP lets developers use the encoding rules which they are most familiar with. [9]

-**SOAP RPC** defines how to generate remote procedure calls via SOAP. A procedure is a set of instructions that tells the specific component how a task will be performed. This can be that the component has to execute methods with data that was received and obtain a certain result. A procedure call includes which method will be invoked and if there is any parameters for the method, the procedure call carry them as well. The technology lets applications invoke methods of a component residing on another machine. It is in this way the applications communicate with the methods of the XML Web Services, and SOAP RPC is the mechanism that enables it. [9]

2.4 WSDL – Web Service Description Language

Before an application can access a Web Service, it has to make sure to locate the right Web Service for the specific assignment. The Web Service has to be able to perform the task which the application requests. The application must learn about the Web Service's capabilities. To obtain the specific information, the application can seek in a registry where all the available WSDL documents are accessible. The registry contain technical information about the Web Services and with these the application can query the Web Services and see if there is one for the specific task. All the information about the Web Service is marked up with XML. [9]

As the name implies, WSDL is created to describe the XML Web Services. It works as a template for describing the XML Web Services. In a WSDL document an XML Web Service's interfaces, data type information, binding information about the transport protocol and how to connect and communicate with the specific Web Service are described. Further, WSDL describes which functionality a Web Service has to offer. WSDL works as a contractor between the service requestor and the service provider. It is both platform and programming independent because it is based on XML and usually describes SOAP services.[10]

2.5 UDDI – Universal Description, Discovery and Integration

UDDI is a specification that defines registries in which companies can registry and add information about themselves and the XML Web Services they provide. The UDDI registry stores the locations of

WSDL documents. To publish a Web Service, the company or developer registries the WSDL document derived from a Web Service at the UDDI registry. The procedure of finding the inquired Web Service is that a service consumer discovers the WSDL document in the UDDI registry and verifies that the Web Service is right for the actual assignment. Then the consumer can access the Web Service via SOAP messages.[9]

With UDDI, every resource that can be accessible via the Internet is published. The one that is responsible for the Web Service registries information about the Web Service in the UDDI registry. It offers advantages to both Web Service providers and Web Service consumers if the company's Web Services is registered in the UDDI registry. For the company, the registry works fine for advertising the Web Services. Because the UDDI registry can be accessed from anywhere, the company acquires global visibility, it gives them the opportunity to communicate and improve their relationship with companies all around the world. In this way, the company gets a wider market and ties new economic bonds with global companies. For the Web Service consumer, the UDDI simplifies the process finding a specific Web Service. As UDDI stores technical information about Web Services, a consumer does not have to spend time searching for aspects regarding; how to connect to the Web Service or what kind of methods the Web Service can execute for example. All this is presented in the UDDI registry.[9]

2.6 Security

It may not seem obvious why the security regarding XML Web Services stands before such a challenge. The technique uses, as described earlier in the report, SOAP which in most cases is bound to HTTP as its transport protocol. HTTP has been able to have assistance from SSL to enable a secure communication. In addition, web authorization tools already exist on the market. [8] In the following paragraphs, some arguments why the message should be responsible for the security of the communication and not the transport layer will be presented.

Sensitive information such as credit card numbers can be sent openly via the Internet. When customers purchase merchandise on e-commerce sites, personal information subscribes on web sites and can be accessed by people who are not authorized. There has to be a way to make sure that spiteful hackers and attackers cannot access the information. A common technique to use when sending sensitive data from one point to another is to adopt the strength of SSL. SSL works at the session layer in the OSI model to provide point-to-point confidentiality and one-way or two-way authentication. The channel that is produced between two parties that are using SSL is secure. When using SSL the data is encrypted when it is sent from the web browser and decrypted when it reaches the web server. As in our business-to-customer (B2C) scenario described earlier, it is important that the customer knows that when he/she enters the credit card number and submits the purchase that the details are securely sent. This sounds like a powerful technique and the obvious question that is popping up; Is not SSL enough? But consider this. If the business that receives the credit card number uses other Web Services for getting details about the customer, maybe to control that the credit card is valid and another Web Service for ordering the products for the customer. In this case, the business handles sensitive information, especially the credit card number and maybe there are some sensitive information on the quotation that is sent to the other helping Web Service. The multi-hop topology that this B2C scenario executes implies that a single message may stop at several Web Services before the task is accomplished. Using SSL, the message is encrypted as it passes between Web services but it will be unencrypted between hops when it reaches the nodes. This allows unauthorised users to read and make undetectable changes to the message when it is between hops. This is because after the decryption phase, the message is in plain text and unauthorised users can access the sensitive information. That is why SSL is not enough in these situations. Used alone, SSL does not provide complete protection for the environment where more than one Web Service is involved. It is also hard to prove that an SSL session ever existed when it is closed.[16][8] There has to be other techniques that make the communication secure even if the message are passing more than one point. In contrast to this, message level security technologies can provide end-to-end security in Web Services environments. If one makes the message responsible for the security, the message can travel between several intermediaries and use different protocols in the communication. This is why XML Encryption, which will be introduced later in the report, is an excellent alternative for SSL. It enables data to be encrypted at the message layer which secures the message's confidentiality between hops and it can ensure that no undetectable changes are made to the message when the message reaches the intermediaries.

Further, in a client/server solution, the server must be able to trust the clients. The client that tries to connect to the server assumes to be honest with its identity. The client must be able to be authenticated before a communication can establish. If one places a user name and password in the header of a SOAP

message, the server can verify if the client has the permission to access the data and functionality that offers at the server.

Working with XML Web Services; the data is usually transferred over the HTTP protocol via port 80 and this one is mostly open for the world around. It is easy for an attacker to view the information. XML Web Services move transactions outside firewalls and enable outside malicious attackers to access applications, giving them access to sensitive information. That is also why XML Web Services present new security challenges.[9] There is a limitation using HTTP as a transport protocol. When one uses HTTP for sending messages between an application and XML Web Services one can authenticate the caller, sign the message and encrypt the contents of the message. This can be done using existing standards such as digital signatures, digital certificates or with SSL as mentioned above. This makes the communication secure in many ways; the application that tries to access the XML Web Service is known, the receiver of the message can control that the message was not tampered and the attackers can not figure out the contents of the message. Still, the security that HTTP offers is not enough. HTTP combined with SSL can only assure security from one point to another. When XML Web Services come into the picture, messages are sent along a path more complicated than not only from one point to another but even over a transport protocol that does not have to do with HTTP. SOAP is located in layer 7 in the OSI model, same as the protocols HTTP and SMTP. But as described in earlier paragraph, SOAP can use HTTP for one leg of the communication and SMTP for the other one for example. The communication must be secured between multiple intermediaries where different transport protocols can be used and the security must be guaranteed all the way, end-to-end.[2]

2.6.1 Authentication

Authentication is the process of verifying that someone is who he/she/it claims to be. This could be all from a user that tries to login on an e-commerce site to when an application tries to communicate with an XML Web Service. The application that seeks communication is known as a principal and the evidence that the application shows is called credentials. In this case the application can bring a username combined with a password as credentials and if they are correctly combined the communication between the entities starts.[7] As in the daily life, a person is using its credentials frequently for verifying that he/she is the right person. For example, when the person uses its credit card for buying items or crossing a country's border showing its passport; the credit card is used to identity the person to the credit agency and the passport vows for the person's identity.[2]

Effective XML Web Services security mechanisms must be able to allow clients to access appropriate services and keeping sensitive information confidential. To access a secured XML Web Service, clients must provide some form of authentication, such as user name combined with password or stronger authentication, such as digital signatures and/or certificates.[9]

2.6.2 Authorization

When the client that is trying to communicate with an XML Web Service is authenticated, the next part of the process is authorization. The security mechanism regarding XML Web Services must be able to authorize certain clients to specific domains of the XML Web Service's functionality. Some clients may not have the permission to access certain methods in the XML Web Service for instance, while other may have full permission and can execute all the methods within the XML Web Service.

2.7 Security Threats

The security threats that the technology XML Web Services is standing before are threats that have been targeted the IT industry in the past. Most of the threats are in fact old threats, such as buffer overflow attacks that tries to trigger specific actions at the attacked computer, for example, to delete the files, change data, or reveal classified data. The attacker tries initially to send more data than the targeted computer expects. After the web server has dealt with the data, it is stored in the memory and maybe executes illegitimate commands at the server. But the attack at SOAP is new for the industry. The security at the application layer has been limited in the firewalls but has progressively moved up the OSI model to reach the application layer. The attacks that focus on individual web applications and Web Services are the ones that a company has problem to discover and provide protection against.[8] Here are some examples of attacks:

SQL attacks. The attacker tries to insert SQL statements in a SOAP message that is sent to an XML Web Service. The attacker hopes that the returning SOAP message will include data from and information about the database, which the Web service is connected to.[8]

Directory traversal attacks. The attacker tries to find SOAP services that maybe are not globally

offered. The attacker can try to manipulate an existing address to a Web Service just to find and use services that the user is not suppose to use.[8]

URL string attacks. The attacker tries to feed inappropriate data into the Web Service to get unexpected result or return more information from a database. The parameters that the SOAP message is expecting is easy to find in its WSDL file. For example if a search SOAP service takes an integer between 1 and 10 as a SOAP parameter, what kind of result is the attacker returning if 1000 is submitted?[8]

2.8 Web Services Security (WS-Security)

Lack of security is the biggest issue that slowing down the progress with enterprise adoption of XML Web Services, according to a study conducted by Hurwitz Group.[9] If a new technology like XML Web Services will stay on the market for a longer period there has to be a way to guarantee the security in the communication between those involved. A company that stands before adoption of XML Web Services in their business and development process must be sure that they can guarantee the security for their customers and in their systems. It is particularly important for a company that operates with transactions containing sensitive information about themselves and their customers. It could be all from organisations number to sensitive information for the parties involved in the communication. Two parties must be able to obtain a secure communication without being threatened by malicious attackers.

In April 2002, Microsoft, IBM and VeriSign released the WS-Security, their joint plan solving the problems described in the last paragraph.[17] WS-Security is a specification that designates using existing specifications and standards for achieving security when a company adopts XML Web Services in their development process.

WS-Security is the first of the security specifications to be released as part of the Web Services security road map stated by Microsoft and IBM. The future specifications are shortly described in Appendix B. Exploring Appendix B, one can see that the specifications are produced bottom up. SOAP is at the base of the figure in the Appendix B. There is nothing under SOAP, because SOAP is transport independent. WS-Security is placed above SOAP and it provides an approach for encrypting and signing SOAP messages, using XML Encryption and XML Signature respectively. WS-Security also endorses to implement security tokens in a SOAP message to represent a user's identity.[8]

A big advantage according to WS-Security is that the SOAP message is responsible for the security in the communication, not the transport protocol. The SOAP message usually travels over HTTP because in that way the message reaches behind the firewalls. However, if one sees it from the developer's point of view, it is the developer's responsibility to make it happen. That is the negative part of WS-Security; the developer has to come up with brilliant solutions that can guarantee the security.[4]

The WS-Security specification recommends ways to extend the security in SOAP messages that can be used to build secure XML Web services. WS-Security is flexible and is designed in that way that the developer can use the security mechanism that he/she is used to. Examples of security technologies that the specification brings together are Public Key Infrastructure (PKI), Kerberos, and SSL. WS-Security provides support for multiple security tokens and multiple encryption technologies such as XML Encryption and XML Signature for signing the document.[3]

To achieve security in the communication between the parties involved, one has to consider the following; the message should reach the receiver without being modified, outsiders shall not be able to read the content of the message while in transit and the sending party should be able to proof its identity. In WS-Security, these requirements or goals are equivalent to message's integrity, message's confidentiality and usage of a security token. These mechanisms can be used independently one by one; one can use only an encryption technology for securing the context in the message. Alternatively, one can combine all three technologies in a tightly integrated manner for achieving a high level of security. This enables XML Web Services providers to develop their own solutions that meet their security requirements of their applications.[6]

2.8.1 XML Encryption

An application that communicates with an XML Web Service sends and receives SOAP messages that in most cases contain important and sensitive information. Securing the confidentiality of the message in that way that outsiders cannot read the content is important. WS-Security endorses using XML Encryption by encrypting parts of the SOAP message to achieve message's confidentiality.[3]

XML Encryption specifies that part or the entire body of a SOAP message can be encrypted. The encrypted data is placed inside an `<EncryptedData>` element. This element is

referenced form an element in the Security header element.[18]

For the `<EncryptedData>` element, a hint at the encryption key used can be specified in the `<KeyInfo>` element, and the encryption algorithm can be specified in the `<EncryptionMethod>` element.[18]

The data is encrypted and outsiders cannot access the data without knowing the technology behind the encryption. XML Encryption can combine symmetric and asymmetric cryptographic technologies. With symmetric encryption of the message, the client and the requested service provide an identical key. This key is used to both encrypt and decrypt the entire message or specified data. When using asymmetric key pair on the other hand, the client and the service have one private key each. They have also a public key that resides at both parties. The sender can encrypt the data with the public key and when the message reaches the calling party, it uses its private key to decrypt the message. These two techniques can be combined; the content can be encrypted with the symmetric key, the symmetric key is encrypted with a public key and both the encrypted content and the encrypted symmetric key are then sent to the receiver.[19]

The distribution of the symmetric keys is a problem for a company. The company must have access to both the client and the service that the client communicates with. This could be possible in an Intranet solution where there is no problem to add the symmetric key at both the server and the client side. But if the client application resides on a place outside the Intranet, the company can combine the key distribution with both a private and a public key. [19]

2.8.2 XML Signature

To verify that a message reaches the calling party without being modified one can contribute with a technique that is widely used, namely digital signature. The WS-Security specification endorses to use XML Signature when signing a SOAP message. XML Signature can be used on XML document and on other digital documents. The message's integrity is guaranteed when a developer uses XML Signature.[3] It is important for both the client and the XML Web Service to know that data has not been modified. In that way, the client can trust the communication and be willing to send sensitive information. This is an argument for a company that adopts XML Web Services in their software development. Their customers must be sure that the solution ensures that data can be sent without being modified.

The technique behind digital signatures uses an asymmetric key pair to verify the sender of the document. The procedure to obtain the digital signature is as follows; a checksum is calculated on the basis of the information that will be signed. This checksum is then encrypted with the sender's private key. The encrypted checksum which now works as a digital signature is attached with the original message. The content of the message which maybe is encrypted and the digital signature can now be sent via the Internet. When the message reaches the receiver, the digital signature is decrypted with the public key. To verify that this is the right signature, the receiver makes the procedure again; calculates a new checksum with the same formula on the signed information. The signatures are compared to each other and if they are identical, the receiver knows that the sender is the right one and that the information has not been modified while in transit.[20]

2.8.3 Web Services Enhancements 1.0

Web Services Enhancements (WSE) 1.0 is a toolkit, which enables for a developer to implement secure XML Web Services on Microsoft .NET platform according to WS-Security. The toolkit provides a powerful supplement when adding a signature on a SOAP request for instance or when encrypting the SOAP response.[21]

WSE for Microsoft .NET is a .NET class library that makes it possible for a developer to extend the functionality of XML Web Services. Also the functionality of the clients, which consume the Web Services, can be extended. After installation of WSE, a developer can implement classes derived from the namespace `Microsoft.Web.Services`. It is those classes a developer has to use to extend the functionality and implement a solution according to WS-Security.[4]

WSE provides a powerful programming model, which lets a developer to manipulate SOAP messages. The programming model makes it possible for a developer to capture the SOAP message both before it leaves and before it is delivered at an XML Web Service and a .NET client respectively. This functionality is enabled by two sets of filters. All SOAP messages leaving a process are processed by one filter called output filter and there is a filter called input filter, which processes all the messages that arrives to a process.[4] A process can be as described earlier both an XML Web Service and a client that consumes the Web Service.

If a developer uses WSE when building an XML Web Service, WSE or a compatible toolkit must be

used as a supplement in the implementation of the client that shall consume the Web Service. That is a small restriction with WSE. If a company wants to interact with a company using XML Web Services, both of the companies must agree of using WSE. In this way it restricts the use of WSE to only integrate applications developed with the help from WSE. But consider the security regarding an XML Web Service environment. Since WSE has support for message-based security, adopting a toolkit like WSE that enables creation of a secure communication is a small price to pay.[4]

3. Method

The project chose to introduce a broad background that presents the technology XML Web Services and its underlying standards from a rather open point of view. The intention was that the person that is responsible for adopting XML Web Services in a similar company like Fishbone Systems AB wants a background that is easy to grasp. In this way, the implementation and understanding of this technology is easier. Still, the background is limited because parts are selectively chosen to illuminate those areas that are important in this case. In addition, the background would have been too complex to understand if it had been too broad.

The following sections motivate why the project choose the one approach that was finally implemented. Further, the technical aspects that have been important throughout the project are declared.

3.1 The practical implementation

The goal was to implement a solution where an XML Web Service could communicate with an ASP .NET application in a secure way. The implementation was supposed to illustrate those criterions that were important for Fishbone Systems AB when developing secure XML Web Services. The two main issues that were stated by Fishbone Systems AB were:

- Encrypt the SOAP response with XML Encryption to secure the message's confidentiality. In this way outsiders will have problem viewing the content of the SOAP message that is transferred.

- Introduce the use of a security token that proves the end user's identity. A user name and password should be used to authenticate the user. These credentials should be verified in a database.

The implementation should be produced according to the WS-Security specification. After discussions with Fishbone Systems AB, the suggestion that the project decided to implement was a fictive credit account service. A more detailed description of the credit account service will follow in next chapter.

3.1.1 Waterfall model

As practical implementation method the Waterfall model was implemented in the project. As the name implies the process is described as a waterfall divided in different phases. Each of the phases; requirements definition, system and software design, implementation and unit testing, integration and system testing, operation and maintenance must be finished before starting on the next phase.[22]

The disadvantage with the Waterfall model when adopting it in the development process is its inflexibility. The distinct phases can cause problem when collaborating between phases. It can be difficult to respond to changes requested from the client. Therefore, the Waterfall model should only be used when the requirements are well documented and understood.[22]

Because the lack of any real clients to discuss the requirements during the developing process the Waterfall model was suitable in this project. The requirements were never changed during the development process. The phases were also completed in its original order; there was no need of starting of one phase before completion of the other. Obviously, information and the phases overlap with each other, but this was according to the specification of the Waterfall model.[22]

3.2 WSE support

The project evaluated the classes derived from `System.Web.Services.Protocols`, and developed a secure implementation using this approach.[23] This approach was later rejected because it is not supported by WS-Security. This approach could be seen as an alternative approach to WS-Security. To implement the SOAP extensions, which led to the secure solution at Fishbone Systems AB the project decided to work with WSE instead. This is because WS-Security recommends for securing the confidentiality of the SOAP message, the use of XML Encryption is best practise. WSE can be used to empower the solution with XML Encryption.[3] Fishbone Systems AB also encouraged the project to choose this alternative because they wanted an implementation based on WS-Security and use the benefits of WSE.

3.2.1 Technical aspects

The practical implementation was based on the following requirements/prerequisites:

- The Integrated Development Environment (IDE) was Microsoft Visual Studio .NET 2003 v. 7.1.3088. The choice fell on this IDE because Fishbone Systems AB develops mostly of their software in this one.
- Microsoft .NET Framework 1.1 was preinstalled. This is a prerequisite for running the Microsoft Visual Studio .NET and necessary for the developer to be able to make use of the functionality that derives from it.
- Web Services Enhancements 1.0 was installed. This was done to obtain the extended functionality these classes bring to the developer. After installation, the developer can capture the SOAP message that is sent between the client and the XML Web Service similar to SOAP extensions.
- Visual Basic .NET (VB .NET) was the programming language that was used. Fishbone Systems AB develops the absolute majority of their software in this programming language. That is why VB .NET was used in this project.
- Active Server Pages .NET (ASP .NET) was used to implement the client application. The code behind derived from the VB .NET. Fishbone Systems AB develops their web applications using ASP .NET and that is also why it was adopted in this solution.
- Microsoft SQL Server 2000 was used as database in the solution.
- When the XML Web Service returned a pure XML document, the usage of Extensible Stylesheet Language Transformations (XSLT) was adopted at the client side for converting the XML data to HTML. XSLT is W3C's recommendation for transforming XML data to HTML.
- When it comes to the implementation of the XML Encryption, the project made a choice between symmetric encryption and asymmetric encryption technique. In this project, the symmetric encryption technique was implemented. With symmetric encryption, it requires that the client and the requested Web Service share the same secret. That is, the key that is used to encrypt the message is the same key that one uses to decrypt the content of the message. Symmetric encryption is a usable technique if one controls both the client side and the server side because one then eliminates the problem with the key distribution. To use the symmetric encryption properly the keys that are used in the encryption/decryption process have to be placed at the client and the server side. In Fishbone Systems AB's case, this is not a problem. Because they have the possibility to place the keys at both endpoints, without have to send them via the Internet. They control both endpoints, in that sense, that they can place each of the key locally. One person of the personnel will simply install the symmetric keys at the client and at the server side. If Fishbone Systems AB did not control both endpoints, the project would have chosen asymmetric encryption instead. Working with X.509 certificates one does not have to consider the key distribution problem. The endpoints sending and receiving the data can publicly post its public certificate and allow anyone to encrypt information using this public key.[2] The receiver is the only one that has the possibility to decrypt the content of the message with its private key. In other words, a secure communication between the server and the client takes place and only the desirable client understands the message.
- The threat that the message can be changed while in transit is Fishbone Systems AB not that exposed to. According to Jörgen Bjerkesjö at Fishbone Systems AB, the company's interest for now and the primary focus is to secure the message's confidentiality. To protect the message's integrity is a future question. That is why XML Signature that had been an alternative for securing the integrity of the message was not used in this implementation.
- SSL was used between the ASP .NET application and the end user's web browser to enable a secure connection. In this way, the confidentiality of the message was still protected.

4. Part 2

4.1 Credit Account Service

In the following sections, the fictive Credit Account Service (CAS) that was implemented is described. The focus is on how the secure communication was enabled. First the web application's functionality and why it is useful for a customer will be described in short. The CAS was accessible via Fishbone Systems AB's Intranet. The connection

between the web browser and the client application was enabled with SSL.

The CAS should simulate a common problem for a credit card company. The company should have a service where their customers easily could control their purchases. This service would definitively be accessible via the Internet, which makes it easy for a customer to reach it. It is there the problem arises. The company has to be able to ensure the security when they are handling with sensitive information such as a customer's credit card number and personal details. It is, as described throughout the report, difficult to ensure the security when working in an XML Web Service environment on the Internet. The CAS's comes up with an approach to ensure the security in a scenario like this. The CAS's main purpose is to give the customers a view over their purchases in a trusted and secure way.

4.1.1 Use Cases

These use cases illustrate the scenarios where a customer can interact with the CAS.

4.1.1.1 Log in

To access the CAS, a user has to log in with its credentials. The credentials consist of a user name and a password. The credentials are then sent to the web application via HTTPS which enables a secure connection between the involved parties. Then, the web application makes the SOAP request by adding the user's credentials in the header of the SOAP message. This request is sent to the XML Web Service. After the XML Web Service has processed the request, two scenarios can occur:

1. The user that tries to log in is not a customer in the CAS. The user has then the possibility to re-enter the login details or the user can go through the registration procedure and fill out the registration form.

2. The user is a customer in the CAS. Then the XML Web Service will return the account details to the user. The details are securely sent, in that sense that the message's confidentiality is secured because the message is encrypted. The authentication is also performed by verifying the user name and password in a database.

Detailed description and illustration of the steps when a SOAP request and SOAP response respectively is performed can be found in Appendix C.

4.1.1.2 Registration

The registration in the CAS is as simple as it can be. The future customer enters user name, password, given name and surname in the form and presses submit button to perform the registration. If the implementation should be implemented in real life, the registration form might have included more personal details and details about the credit card.

4.1.1.3 Make purchase

After the user is logged in, the user can make a fictive purchase. The user has to fill out a form containing details about the purchase. The details are purchase date, store and amount of purchase. If the purchase succeeded, the account details are updated and the details are once again securely sent to the client from the XML Web Service.

4.1.1.4 Get account details

The user can return its credit account details by pressing a button. This is the same procedure as when the user is successfully logged in. The XML Web Service returns all the account details that belong to the specific user.

4.1.1.5 Pay unsettled records

The user can choose to pay all the unsettled records. This is also a fictive activity. It is a text next to each record that indicates if the record is paid or not. By pressing the pay button, the user accepts to receive an invoice with all the unsettled records. The records are then updated and returned to the user once again. It is now a text next to each record that indicates that all the records are paid and updated.

4.1.1.6 Log out

The customer presses a log out button for performing a log out.

4.1.2 Implementing WSE in CAS

When using WSE in the development process of an XML Web Service solution, the developer has to make a few additional configurations in the development environment. The adjustments are depending on what kind of functionality the company wants to extend the solution with. In Fishbone Systems AB's case the focus was on encrypting the body of the SOAP response and to

implement a user name token. The following adjustments had to be made for enabling a secure XML Web Service solution:

- When building a solution where a client consumes an XML Web Service, the reference to the `Microsoft.Web.Services.dll` must be added both at the client and in the XML Web Service project. This reference was added to both the client application and the XML Web Service in the CAS.
- The proxy class that works as a middle layer between the client application and the XML Web Service must be modified. There are two techniques that a developer can use when adding the proxy class to a client application. The first one is to use Visual Studio .NET and add a Web Reference. In this way, Visual Studio .NET creates the proxy class for the developer. The second alternative and the one that this project adopted is to create the proxy class with the help of the wsdl.exe command. This command allows one to specify for example what programming language the XML Web Service is created with and the URL where the XML Web Service is reachable. In this way, one has more control of the creation of the proxy class and one can easily specify different attributes for the command. After creation, one has to include the proxy class in the project.

  To make use of WSE's input and output filters, the proxy class must extend the class in `Microsoft.Web.Services.WebServicesClientProtocol`. This class inherits the default base class for XML Web Service proxy classes, `System.Web.Services.SoapHttpClientProtocol`. When the proxy class is modified, it ensures that WSE's filters have a chance to process the SOAP messages that are exchanged when the client application invokes one of a proxy's methods.[4] In other words, when a client application tries to invoke a method at the XML Web Service. See Appendix D for complete source code over the proxy class that was implemented in the solution.
- The web.config files must be modified both in the client application and in the XML Web Service in order to use the security support WSE brings to the solution. The changes in the XML Web Service's web.config file are illustrated here:

```
<webServices>
      <soapExtensionTypes>
            <add
            type="Microsoft.Web
            .Services.WebServic
            esExtension,
            Microsoft.Web.Servi
            ces,
            Version=1.0.0.0,
            Culture=neutral,
            PublicKeyToken=31bf
            3856ad364e35"
            priority="1"
            group="0" />
      </soapExtensionTypes>
</webServices>
```

Since WSE is implemented as a SOAP extension these lines have to be added to make the WSE registered in the web.config file. These changes are made to add a `Microsoft.Web.Services` element inside the configuration tag in the web.config file. This element controls the configuration of all WSE input and output filters.[3][24] See Appendix E for the complete XML Web Service's web.config file.

```
<configSections>
      <section
      name="microsoft.web.servic
      es"
      type="Microsoft.Web.Servic
      es.Configuration.WebServic
      esConfiguration,
      Microsoft.Web.Services,
      Version=1.0.0.0,
      Culture=neutral,
      PublicKeyToken=31bf3856ad3
      64e35" />
</configSections>
```

These lines are added both in the client application's web.config file as well in the XML Web Service's web.config file. In order to inform WSE of the different functionality that the solution will use one has to make an appropriate WSE configuration setting. In this case the WSE has to be prepared for decryption at the client side and use of a user name token at the server side. This section tells the .NET runtime which class to use to parse the WSE-specific configuration data.[3]

```
<microsoft.web.services>
      <security>
            <passwordProvider
            type="WSE_Security.
            WSEPasswordProvider
            , WSE_Security" />
```

```
      </security>
</microsoft.web.services>
```

In this section the XML Web Service is informed which class that provides the password. `WSE_Security` is the name of the namespace, `WSEPasswordProvider` is the name of the class and `WSE_Security` is the name of the assembly.[24]

The additional changes in the client application's web.config file are illustrated here:

```
<microsoft.web.services>
     <security>
           <decryptionKeyProvi
           der
           type="WSE_SecurityC
           lient.DecryptionKey
           Provider,
           WSE_SecurityClient"
           />
     </security>
     <diagnostics>
          <trace enabled="true"
          input="C:\Temp\inputT
          race.config"
          output="C:\Temp\outpu
          tTrace.config" />
     </diagnostics>
</microsoft.web.services>
```

This section contains two modifications. The changes between the security tags are notifying WSE runtime which class that are returning the decryption key at the client side. Same syntax as when describing the password provider class is adopted here. `WSE_SecurityClient` is in this case the name of the namespace, `DecryptionKeyProvider` is the class name and `WSE_SecurityClient` is the assembly name.[24]

It is possible to store the SOAP messages that are sent between the client application and the XML Web Service. This is accomplished by adding the information between the diagnostics tags.[24] The messages were used in the debugging process. See Appendix F for the complete client application's web.config, Appendix G and Appendix H for a SOAP request and a SOAP response respectively.

4.1.3 Implement User Name Token

This section illustrates the use of the user name token that was implemented in the CAS. The user name token verifies if a customer has the permission to access the XML Web Service. The following code snippets are taken from the file `AccountService.aspx.vb`. See Appendix I for a complete version of the file.

```
Imports
Microsoft.Web.Services.Security
```

At the top of the class the import statements are listed. These lines enable for the use of the `Microsoft.Web.Services.Security` namespace. It is in this namespace the user name token is located. From now on, it is possible to create a user name token in the class.

```
#Region "Private Variables"
       Private oUserNameToken As
       UsernameToken
       Private oWebServiceRef As
       WSAccountService.AccountService
#End Region
```

In this region, the private variables for the user name token and the reference to the proxy class are declared.

```
oUserNameToken = New
UsernameToken(Me.txtUserName.Text,
Me.txtPassword.Text,
PasswordOption.SendHashed)

oWebServiceRef = New
WSAccountService.AccountService

oWebServiceRef.RequestSoapContext.Secu
rity.Tokens.Add(oUserNameToken)
```

These lines are executed when a customer presses the log in button. The user name token is initialized with a user name and a password, which the current customer has entered in the text boxes; `txtUserName` and `txtPassword` respectively. There are three alternatives when applying the password in the user name token. One can exclude it, send the password in clear text and the more secure approach is to send a hashed password. In the CAS the password was sent hashed. This digest version of the password is a combination of the password, a nonce and the creation time. The nonce is a unique string that identifies the specific request.[25] The hashed password (Password_Digest) is created with this formula:
Password_Digest = Base64 (SHA-1 (nonce + created + password))
The formula uses the SHA-1 hash algorithm with the three components, and includes the Base64 encoding on the result.[26]

The next two lines initialize the reference, `oWebServiceRef`, to a new proxy class. It is via this reference the communication to the XML Web Service is enabled.

The last two lines are the important ones when using a user name token. When these lines are executed the user name token is placed in the header of the SOAP request. See Appendix G for a complete listing of the SOAP request. The user name token is represented by `<wsse:UsernameToken>` elements, which are nested inside the header elements. Inside the user name token element, there are elements describing the user name and password; `<wsse:Username>` and `<wsse:Password>` respectively.

When the client application has generated the SOAP request the XML Web Service is responsible for verifying if the customer is a valid customer in the CAS. The input filter in WSE passes the user name token to a password provider, which returns a password. The password provider is a class that implements the interface `IPasswordProvider`, which is located in the `Security` namespace.[4] In the CAS, the customer is verified against a database. The password provider implements a method called `GetPassword`, which acquires a user name token as a parameter. If the password that the method returns matches the password in the header, then it is a valid customer and the customer can access the web method. See Appendix J for a complete representation of the password provider class.

4.1.4 Securing the message's confidentiality

The SOAP response was encrypted empowered by XML Encryption in the CAS.

The encryption and decryption process in the CAS could be generated with two block-cipher algorithms, Triple-DES and AES. These two are both a symmetric encryption technique and they are both supported by XML Encryption. The encryption in the CAS was introduced with Triple-DES. Triple-DES as is a block-cipher algorithm encrypts data block by block. The block size in the CAS was 64 bits. The decryption of a block of ciphertext depends on the preceding ciphertext block. This implies that if a block is changed the following blocks will be affected. The blocks' dependency also mean that if they are rearranged the decryption process will not come up with a correct solution.[8]

The encryption and decryption processes contain an identical symmetric key obviously. The processes also include an initialization vector which is used to ensure that the same plaintext will not result in an identical ciphertext if the texts are encrypted with the same key. There occurs a problem if the initialization vector is changed. Then the decrypted plaintext will also be changed. However, XML Encryption does not protect the integrity of the message, only the confidentiality of the message.[8]

The following code snippets represent the actual encryption and decryption process when a customer is logged in.

```
oXmlDocument =
oWebServiceRef.GetCreditAccountDetails
```

When a call to the web method is performed these lines are executed at the client side. The web method in the XML Web Service returns an XML document. See Appendix K for complete source code of the transformation process of the XML document. The XML document is transformed into HTML using XSLT and then presented for the customer.

At the XML Web Service the XML document will be encrypted before it is sent to the client. The procedure of the encryption is as follows:

```
Dim oResponseContext As SoapContext
oResponseContext =
HttpSoapContext.ResponseContext
```

These lines declare and initialize the `SoapContext` `oResponseContext`. It is via this object one has access to the SOAP message that is returned. In this case the mode is set to `ResponseContext` which indicates the response action.

```
Dim oEncryptionKey As EncryptionKey
oEncryptionKey = New
SymmetricEncryptionKey(oSymmetricAlgor
ithm)
```

When these lines are executed the WSE encryption key is created.[18] `OSymmetricAlgorithm` is in this case the symmetric key that is created. This one is identical in the client application and in the XML Web Service.

```
Dim oEncryptedData As EncryptedData =
New EncryptedData(oEncryptionKey)
```

These lines create an element `EncryptedData` with the WSE encryption key.[18]

```
oResponseContext.Security.Elements.Add
(oEncryptedData)
```

The encryption key is added to the SoapContext element for the response message and is used by

WSE's output filter to encrypt the XML document that the method returns. See Appendix L for complete source code representing the XML Web Service. See also Appendix H for the complete SOAP response that is created, where the body is encrypted.

When the encrypted SOAP message is received at the client application, WSE's input filter calls a decryption key provider on the client side to receive the symmetric key.[18] The class has one method, `GetDecryptionKey`, which returns the decryption key. The key that is returned is identical to the one in the XML Web Service. It is this key that is used to decrypt the content of the message. See Appendix M for a complete source code of the decryption provider class at the client side.

5. Result

WS-Security is a good alternative for a company to apply when developing secure XML Web Services. The WS-Security specification provides guidelines for the company on how to guarantee the security in the communication between applications and XML Web Services. For a company that deals with sensitive information such as customer's credit card number for instance, the security is of high importance. The company must be able to guarantee for its clients and customers that they can submit sensitive information on the company's web site. Because the technology XML Web Service specifies that data embedded in SOAP messages usually are transferred over the HTTP protocol, malicious hackers can sniff (intercept) the communication and read the content of the message. According to WS-Security, the security is placed on the message and not on the transport protocol. In this way a message can pass several intermediaries and over different transport protocol and still be secured.[3] The security support, which WS-Security enables, is as follows:

- **Message confidentiality**
  Protecting a message's content from being intercepted by outsiders is a primary security concern which WS-Security solves. WS-Security endorses a company to use XML Encryption when encrypting parts or the entire body of a SOAP message. Applying XML Encryption will secure a SOAP message's confidentiality through the complete transmission.

- **Message integrity**
  To ensure that a SOAP message is transmitted without being modified, WS-Security endorses a company to use XML Signature. The message's integrity is guaranteed in this way.

- **Security token propagation**
  WS-Security supports multiple security tokens and the user name token that was implemented in this project is one of them. Security tokens can be added in the header of a SOAP request, containing credentials from the end user. The requested XML Web Service has then the responsible to authenticate the end user. In this way, the end user's identity is proven in that sense that it is authenticated and authorized to access the XML Web Service.

WS-Security is a useful approach for different companies because of its flexibility. Companies with different needs of security can stick to the same specification. The specification makes it possibly for a company that have the focus on securing the message's confidentiality to make use of XML Encryption. Another company might be exposed to the threat that their messages can be changed while in transit but the confidentiality of the message is not that important to protect. In that case the company can apply a digital signature like the XML Signature, which WS-Security endorses to use for securing the integrity of the message.[3]

To further prove that WS-Security is a good alternative for a company to adopt when implementing secure XML Web Services this degree project has made a practical implementation according to WS-Security. The final solution was a fictive credit account service, which illustrates typical problems a company stands before when achieving security in their XML Web Services. The implementation supports encryption of the SOAP response and also makes use of a user name token, which verifies the client's identity. The practical solution was implemented at Fishbone Systems AB. Fishbone Systems AB is exposed to the threat that outsiders can view the content of the message that is transferred from an XML Web Service to a client application. As according to WS-Security the encryption process in the credit account service was empowered with XML Encryption. In this way the message's confidentiality is guaranteed and Fishbone Systems AB can send sensitive information without being threatened that outsiders can take part of the communication. Appendix H illustrates the encrypted SOAP response. The body of the message is encrypted and the encrypted data is placed inside `<EncryptedData>` elements.

To verify that a user is a valid customer in the credit account service, there has to be a way for the XML Web Service to authenticate the SOAP request. This was solved by leverage the power of a user name token that is placed in the header of the

SOAP request. The user's credentials containing a user name and a hashed password were inserted in the header. This procedure is designated to make the client application authenticated at the XML Web Service and the user authenticated as a valid customer in the service. See Appendix G for a complete listing of the SOAP request. Inside the header elements a user name token is visible.

For a comprehensive illustration of how Fishbone Systems AB's architecture could look like after this project's approach is implemented, the report refers to Appendix N. The credit account service could be seen as one system amongst others.

6. Discussion

When a company adopts XML Web Services in their development process the security regarding the technology is a big concern. If the company contributes to WS-Security the security concerns can be solved. The specification can help the company to protect the message's confidentiality and integrity. It also illustrates different security tokens that the company can implement. The specification is flexible and the company can use the techniques that meet their requirements. If the company controls both end points; XML Web Service and the calling application, then WS-Security is straight-forward to apply. However, a problem might arise when a company adopts the specification. If the company consumes XML Web Services that are developed and controlled by other companies there has to be an agreement between them. The arrangement must specify that both parties have to follow WS-Security. Both parties have to implement Web Service Enhancements to be able to make use the security support WS-Security brings to the solution.[4]

Using HTTP in combination with SSL for example, one can guarantee the security from one point to another. However, when working in complex XML Web Services environment a message can pass several intermediaries and consume different XML Web Services before reaching its final destination. It may also travel with different transport protocol, for example SMTP. In this case the solution created with SSL is not enough.[2] If one instead places the responsibility of the security on the SOAP message one can draw a conclusion that the security is guaranteed through the complete transmission, end-to-end. That is why message-level security that WS-Security brings to the technology is worth to adopt.

The fictive credit account service that was produced in this project is only one implementation of the WS-Security specification. The project's approach was based on WSE when securing the XML Web Services according to WS-Security. However, there are other implementations to apply when securing XML Web Services according to WS-Security. VeriSign offers an open source toolkit, Trust Service Integration Kit (TSIK), to help developers integrate security into XML Web Services according to WS-Security. The implementation provides developers code they can use to achieve higher levels of trust and security in their XML Web Services. The intention is to accelerate widespread adoption of XML Web Services by publish a toolkit that can make them secure. According to VeriSign, companies will not implement XML Web Services until there is an industry standard for securing them.[27] IBM has another toolkit, WSTK 3.3.2, which enables WS-Security functionality for Java. It is possible to integrate this implementation with WSE. The interoperability between WSE for Microsoft .NET and an XML Web Service developed in Java is described in an article conducted by Simon Guest.[28] This indicates that there are several implementations of WS-Security on the market. These could provide applicable security support according to WS-Security.

If Fishbone Systems AB chooses to apply the concept in this project's implementation, their XML Web Services will be secured. According to the implementation, a client application will be authenticated at the XML Web Service which entails that only trusted clients can communicate with an XML Web Service. Additional, the SOAP message's confidentiality is secured in the response because the content of the body is encrypted. This prevents outsiders to take part of sensitive information that is transferred.

7. Future work

For achieving all the three goals specified in WS-Security; securing the message's confidentiality, securing the message's integrity and make use of a security token the credit account service would have to implement a digital signature for securing the integrity of the message. After the digital signature is applied, one can verify if the message has been tampered while in transit. This could be a future addendum.

"*There are specifications appearing on the scene that attempt to secure different facets of Web Services. As each specification becomes standardized and viable over time, the operation of Web Services will be better protected*."[29]

The specifications that the sentences refer to are shortly described in Appendix B. The road map and

security stack was stated by Microsoft and IBM, and in a near future these specifications will be standardized. How will the industry look like after the specifications are standardized? Will it be a revolution in developing secure XML Web Service when the entire Web Service stack is standardized? These are questions that a future research could answer. Further, a future work could be to investigate the other specifications and theirs role in the road map. As a practical implementation, a trusted solution according to the proposed specification WS-SecureConversation could be created to verify that the specifications together are a powerful framework that guarantees the security in the XML Web Services' world.

WS-Security was introduced for the industry in 2002 but has not yet become an industry standard. The specification is delivered to OASIS (Organization for the Advancement of Structured Information Standards) and is on its way towards ratification.[30] While OASIS tackles the job of evaluating WS-Security, other actors on the market consider joining the development of WS-Security. Sun Microsystems intends to have WS-Security support in their Java Web Services Developer Pack (Java WSDP) scheduled for fall 2003. This release will support both XML Signature and XML Encryption.[29] Sun Microsystems has also joined WS-I (Web Services Interoperability) for improving the work around WS-Security. WS-I is an organisation that provides guidelines to companies and developers on how to correspond to published Web Services standards.[31] If WS-Security and the other future standards will be as successful as the creators are hoping on, the biggest players on the market have to apply and implement them.

8. References

[1] Fishbone Systems AB (2003, Sep.). [Online]. Available: http://www.fishbonesystems.com

[2] S. Seely. (2002, Oct.). Understanding WS-Security. Microsoft Corp. USA. [Online]. Available: http://msdn.microsoft.com/library/en-us/dnwssecur/html/understw.asp

[3] B. Atkinson., G. Della-Libera., S. Hada., P. Hallam-Baker., J. Klein., B. LAMacchia., P. Leach., J. Mangerdelli., H. Maruyama., A. Nadalin., N. Nagaratnam., H. Prafullchandra., J. Shewchuk., D. Simon. (2002, Apr.). Web Services Security (WS-Security) v. 1.005. Microsoft Corp., IBM Corp., Versign. USA. [Online]. Available: http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

[4] T. Ewald. (2002, Dec.). Programming with Web Services Enhancements 1.0 for Microsoft .NET. Microsoft Corp., USA. [Online]. Available: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/progwse.asp

[5] M. O'Neill. (2003, Aug.). Architecting Security for Web Services. Fawcette Technical Publications., USA. [Online]. Available: http://www.ftponline.com/javapro/2003_08/magazine/features/moneill/page2.aspx

[6] IBM and Microsoft. (2002, Apr.). Security in a Wes Services World: A Proposed Architecture and Roadmap. IBM Corp., Microsoft Corp., USA. [Online]. Available: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/securitywhitepaper.asp

[7] Microsoft Corp. (2002, Feb.). XML Web Services Security: Are XML Web Services Secure?. Microsoft Corp., USA. [Online]. Available: http://msdn.microsoft.com/library/en-us/dnwssecur/html/xmlwssec.asp

[8] M. O'Neil., P. Hallam-Baker., S. Mac Cann., M. Shema., E. Simon., P. A., Watters., A. White. (2003, Jan.) "Web Services Security," Berkeley, California: McGraw-Hill/Osborne., pp.18, 42-57, 166-172

[9]H. M. Deitel., P. J. Deitel., B. DuWaldt., L. K. Trees., (2003, Aug.). "Web Services A Technical Introduction," Upper Saddle River, New Jersey: Prentice Hall.

[10]E. Cerami. (2003, Feb.). "Web Services Essentials," 1st ed., Sebastopol, CA: O'Reilly & Associates, Inc., pp. 3, 10, 16

[11]D. Gisolfi. (2001, Jul.). Is Web services the reincarnation of CORBA?. IBM Corp., USA. [Online]. Available: http://www-106.ibm.com/developerworks/webservices/library/ws-arc3/

[12] C., S. Horstmann, G. Cornell. (2001, Dec.). "Core Java2 Volume II, Advanced Features," 5th Edition., Palo Alto, California: Prentice Hall

[13] C-J. Nordqvist. (2002, Apr.). Skapa ditt första XML-dokument. International Data Group., Sweden. [Online]. Available: http://www.idg.se/webstudio/pub/article.asp?id=78

[14]M. Augustyniak. (2001, Dec.) ”Teach Yourself .NET XML Web Services in 24 Hours,” Indianapolis, Indiana: Sams Publishing.

[15] D. Box., D. Ehnebuske., G. Kakivaya., A. Layman., N. Mendelsohn., H. F. Nielsen, S. Thatte., D. Winer. (2000, May.). Simple Object Access Protocol (SOAP) 1.1. W3C., USA. [Online]. Available: http://www.w3.org/TR/SOAP/

[16] S. Evans., O. Dowling. (2002, Jul.). Is SSL enough security for first-generation Web services?., WebServices.Org., USA. [Online]. Available: http://www.webservices.org/index.php/article/articleview/529/1/24/

[17] J. Wagner. (2002, Apr.). Web Services Gets Security Blanket. Internetnews., USA. [Online]. Available: http://www.internetnews.com/ent-news/article.php/1007861

[18] J. H. Gailey. (2003, March.) Encrypting SOAP Messages Using Web Service Enhancements. MSDN., USA [Online]. Available: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/wseencryption.asp

[19] F. Hirsch. (2002). Getting Started With XML Security. [Online]. Available: http://home.earthlink.net/~fjhirsch/xml/xmlsec/starting-xml-security.html

[20] Encyclopedia Intranetica. (2000). Digitala och elektroniska signaturer., Sweden. [Online]. Available: http://www.intranetica.com/intranetica/kds/signaturer.shtml

[21] R. Jennings. (2003, Jan.) Upgrade to WSE From WSDK. XML & Web Services Magazine., USA. [Online]. Available: http://www.fawcette.com/xmlmag/2003_01/online/webservices_rjennings_01_06_03/default.aspx

[22] I. Sommerville. (2000.) “Software Engineering,” 6th Edition. Harlow, UK: Addison-Wesley

[23] R. Howard. (2001, Sep.). Encrypting SOAP Messages. Microsoft Corp., USA [Online]. Available: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnaspnet/html/asp09272001.asp

[24] K. Aschenbrenner. (2003, May.) Implement Secure .NET Web Services with WS-Security. DevX., USA [Online]. Available: http://portals.devx.com/security/Article/15634/0/page/1

[25] M. Powell. (2002, Dec.) WS-Security Authentication and Digital Signatures with Web Services Enhancements. Microsoft Corp., USA [Online]. Available: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwssecur/html/wssecauthwse.asp

[26] OASIS. (2003, Feb.) Web Services Security UsernameToken Profile. OASIS., USA [Online]. Available: http://www.oasis-open.org/committees/wss/documents/WSS-Username-02-0223-merged.pdf

[27] VeriSign. (2002, Dec.) VeriSign Offers Open Source WS-Security Implementation and Integration Toolkit to Help Developers Integrate Security Into Web Services. VeriSign., USA [Online]. Available: http://www.verisign.com/corporate/news/2002/pr_20021210b.html

[28] S. Guest. (2003, Feb.) Web Services Enhancements 1.0 and Java Interoperability, Part 1. Microsoft Corp., USA [Online]. Available: http://msdn.microsoft.com/webservices/building/wse/default.aspx?pull=/library/en-us/dnwebsrv/html/wsejavainterop.asp

[29] J. J. Heiss. (2003, Mar.) The Future of Web Services Security: A Conversation with Eve Maler. Sun Microsystems., USA [Online]. Available: http://java.sun.com/features/2003/03/webservices-qa.html

[30] A. Bernard. (2003, Aug.) WS Security and Adoption. Jupitermedia Corp., USA [Online]. Avaliable: http://itmanagement.earthweb.com/secu/article.php/3064501

[31] M. LaMonica. (2003, Mar.) Sun joins Web services standards board. CNET Networks., USA [Online]. Available: http://news.com.com/2100-1013-994183.html

Appendix A
SOAP Envelope

This is the main part of the SOAP architecture and it describes the structure of a SOAP message. A SOAP message is quite informally an XML document. SOAP encapsulates data in XML format in the messages that are sent to and from an XML Web Services. A SOAP message consists of a mandatory SOAP envelope, which composes of an optional SOAP header, and a mandatory SOAP body.[9]

The SOAP envelope component requires information that specifies the namespace and schema information of the message. It is the application sending the message that defines the contents of the header and the body, not the envelope.[9]

The header in the SOAP message can contain information about the message, such as parsing and security information for intermediaries that receive the message. Because a SOAP message does not necessary travel directly from the original sender to the recipient, it can take a route between other nodes. For example, if there is a Web Service that performs transactions or payment processing, the header element can add routing information that tells the message to proceed to another node for checking its identity and then move on depending of the node's connections. That node checks the identity and verifies that the message has the authorization to proceed to the final node. In the header, the developer can increase the functionality of the SOAP message by adding elements in the header, in this case elements for authorization and authentication.[9]

The last part of the message is the body. Here the data is stored, in other words where the purpose of the message is presented for the intended receiver. The body can contain pure data that the receiving application demands or instructions that the actual application will perform. In fact, the Web Service can invoke a method just as easy as if the distributed application was resided on the same machine. In other words, the body contains remote procedure call for the Web Service. After a RPC is executed, the Web Service sends a SOAP message containing the result from the procedure call back to the calling application.[9]

This is an example where a client application makes a request to an XML Web Service. The procedure call for the service is in this case a simple HelloWorld method. The SOAP message that is sent to the XML Web Service is this one (The message is inteded for readability):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Header>
        <mySoapHeader xmlns="http://tempuri.org/">
             <UserName>user1</UserName>
             <Password>passwd</Password>
        </mySoapHeader>
   </soap:Header>
   <soap:Body>
        <GetHelloWorld xmlns="http://tempuri.org/">
   </soap:Body>
</soap:Envelope>
```

The first line states that the message is xml based. The next line starts with a declaration of the Envelope and it includes three namespaces. In this example, the SOAP message includes a Header but as stated earlier, this is not necessary. The header is optional. The header has a name, `mySoapHeader` that contains two properties; `UserName` and `Password`. This client sends a user name and password to the XML Web Service and the XML Web Service can validate that the client has access to the requested web method. This model makes it secure in that manner that the client is trusted if the user name and password is correctly combined. After the Header is closed, the Body starts. In this case, the body only contains the name of the

calling method. It could also include parameters for the method or data to another intermediary. To make this example even more secure one could have encrypted the user name and password in the header and encrypted the body part.

The response from the XML Web Service contains the hello world string. The SOAP message looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
        <GetHelloWorldResponse xmlns="http://tempuri.org/">
            <GetHelloWorldResult>Hello World!</GetHelloWorldResult>
        </GetHelloWorldResponse>
   </soap:Body>
</soap:Envelope>
```

The SOAP message does not include any header. The client added the header to the SOAP message, not the XML Web Service. The XML Web Service on the other hand, accesses the header, validates the user name and the password and if they are correct this message is sent back to the client. The string Hello World is presented between the tags GetHelloWorldResult. The Result part is added to the method name.

Appendix B
WS-Security stack

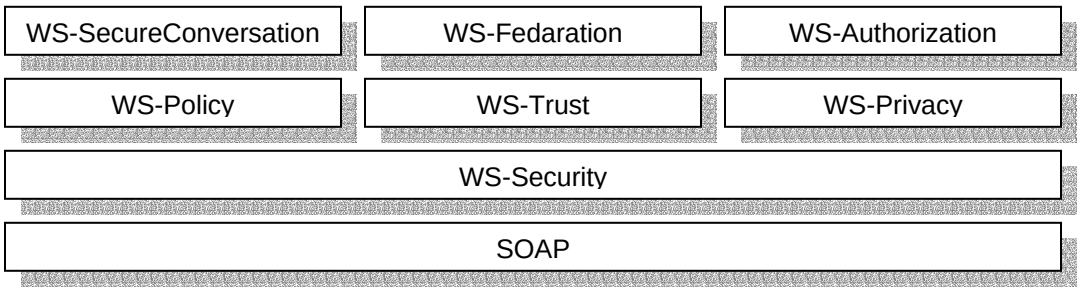


Fig 1. IBM and Microsoft Web Services security specifications.[8]

WS-Policy
WS-Policy allows a company to specify the security requirements of its XML Web Services. These requirements could be represented as a WSDL document. The company can specify if they only consume Kerberos tickets for example.[8]

WS-Trust
WS-Trust specifies how trust relationships are created. A trust proxy can be used to read the WS-Policy information and offer the right security token that is required in the communication. WS-Security will be used to transfer the proper security token. The trust proxy has the ability to add a security token that represent the end user. In this way the SOAP request is originated from the end user.[8]

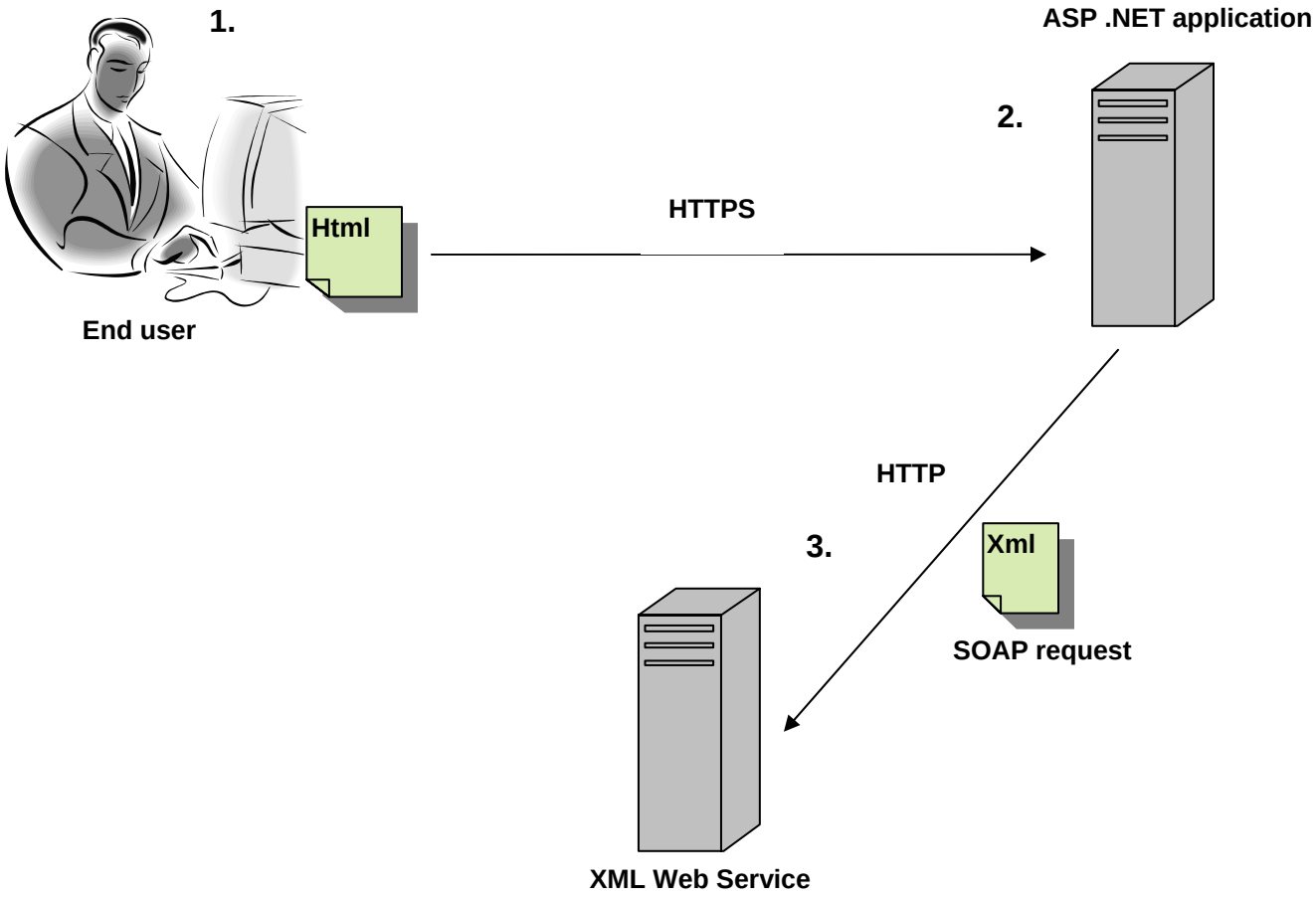
WS-Privacy
WS-Privacy uses a combination of WS-Policy, WS-Security and WS-Trust to communicate private policies. It requires that the incoming SOAP requests contain claims that correspond to these policies. The WS-Security is used to encapsulate the claims in different security tokens.[8]

WS-SecureConversation
When a SOAP message reaches an intermediary, the security token is evaluated and checked against a security policy. This process is time-consuming and a performance issue because it has to be repeated for each incoming SOAP message. WS-SecureConversation solves this problem by allowing a requestor and a Web Service to authenticate SOAP messages and establish a shared authenticated security context. This scenario can be seen as SSL at the SOAP level. WS-SecureConversation is based on WS-Security and WS-Trust in order to negotiate and exchange keys.[8]

WS-Federation
Ws-Federation describes how federated trust scenarios may be created in combination with WS-Security, WS-Trust, WS-Policy and WS-SecureConversation. WS-Policy and WS-Trust are used to decide which security tokens are consumed.[8]

WS-Authorization
WS-Authorization defines how Web Services manage authorization data and policies. This specification is flexible to both authorization format and authorization language.[8]

Appendix C
Request from user



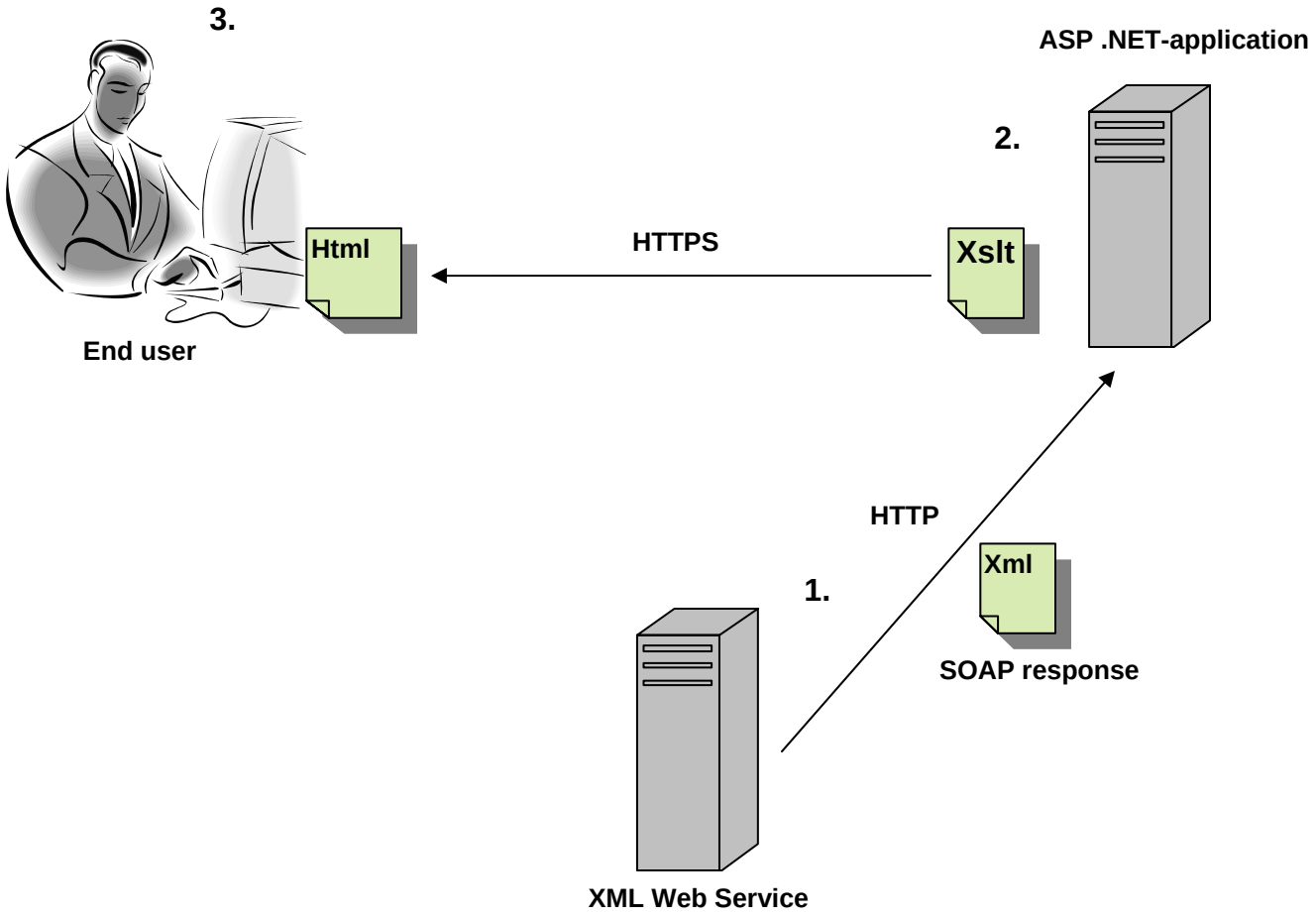
1. In this stage, the end user tries to log in to the credit account service. The end user accomplishes this by entering a user name and password in a form at the start page. The password is entered in a textbox where the mode is set to password. As in most cases where the input form deals with sensitive information, the user is not able to view the password; it is hidden for increasing the security.

   Next, the web browser is using SSL to perform a secure connection to the ASP .NET application. The user's credentials are in this way secured and the message's confidentiality is secured in this leg of the communication. The details are sent via HTTPS to the ASP .NET application.

2. At the ASP .NET application the user's credentials are placed in the header of a SOAP message. The application is creating a user name token, which represent the customer's credentials. It is the application and not the end user that creates the SOAP message and likewise the SOAP request. The SOAP request is sent to the XML Web Service via HTTP. It is a XML based SOAP message that is sent.

3. The SOAP request is treated at the XML Web Service. If the credentials are right combined there exists a user in the database. In this way the customer is authenticated. Then the XML Web Service will return the user's account details in the SOAP response.

Response from XML Web Service



1. The XML Web Service encrypts the body of the SOAP message using XML Encryption. The XML Web Service uses WSE to create an encrypted SOAP response according to WS-Security. The body contains the account details for the specific user that makes the request. The SOAP response that the XML Web Service produces can be sent via several intermediaries and over multiple transport protocols because the responsible of the security is placed on the message.

2. The application decrypts the SOAP response using the same symmetric key as was used in the encryption process. The symmetric key could be updated according to the computer's time for instance. This would increase the security and make it harder for outsiders to come up with an identical key. But there is one problem. The key must be updated both at the same time at both places. And the computers' time must be synchronized as well. When the SOAP message is decrypted the application transform the XML into HTML using XSLT. Then the application is sending the data to the user. The data is sent over HTTPS.

3. The account details are presented in the end user's web browser.

Appendix D
Proxy class

```
'------------------------------------------------------------------------------
' <autogenerated>
'     This code was generated by a tool.
'     Runtime Version: 1.1.4322.573
'
'       Changes to this file may cause incorrect behavior and will be
lost if
'     the code is regenerated.
' </autogenerated>
'------------------------------------------------------------------------------

Option Strict Off
Option Explicit On

Imports System
Imports System.ComponentModel
Imports System.Diagnostics
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.Xml.Serialization
Imports System.Xml

'
'This source code was auto-generated by wsdl, Version=1.1.4322.573.
'
Namespace WSAccountService


'<remarks/>
<System.Diagnostics.DebuggerStepThroughAttribute(),  _
System.ComponentModel.DesignerCategoryAttribute("code"),  _
System.Web.Services.WebServiceBindingAttribute(Name:="AccountServiceSoa
p", [Namespace]:="http://fishbonesystems.com/AccountService")>  _
Public Class AccountService
        Inherits Microsoft.Web.Services.WebServicesClientProtocol

        '<remarks/>
        Public Sub New()
            MyBase.New
            Me.Url                                                    =
            "http://segots0002/MasterProject.WS/AccountService.asmx"
        End Sub

        '<remarks/>

      <System.Web.Services.Protocols.SoapDocumentMethodAttribute("http:
      //fishbonesystems.com/AccountService/GetCreditAccountDetails",
      RequestNamespace:="http://fishbonesystems.com/AccountService",
      ResponseNamespace:="http://fishbonesystems.com/AccountService",
      Use:=System.Web.Services.Description.SoapBindingUse.Literal,
      ParameterStyle:=System.Web.Services.Protocols.SoapParameterStyle.
      Wrapped)>  _
```

```
        Public Function GetCreditAccountDetails() As
        System.Xml.XmlDocument
              Dim results() As Object =
              Me.Invoke("GetCreditAccountDetails", New Object(-1) {})
              Return CType(results(0),System.Xml.XmlDocument)
        End Function

         '<remarks/>
        Public Function BeginGetCreditAccountDetails(ByVal callback As
        System.AsyncCallback, ByVal asyncState As Object) As
        System.IAsyncResult
             Return Me.BeginInvoke("GetCreditAccountDetails", New Object(-
             1) {}, callback, asyncState)
        End Function

         '<remarks/>
        Public Function EndGetCreditAccountDetails(ByVal asyncResult As
        System.IAsyncResult) As System.Xml.XmlDocument
              Dim results() As Object = Me.EndInvoke(asyncResult)
              Return CType(results(0),System.Xml.XmlDocument)
         End Function
    End Class
End Namespace
```

Appendix E
XML Web Service's web.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

      <configSections>
            <section                        name="microsoft.web.services"
            type="Microsoft.Web.Services.Configuration.WebServicesConfi
            guration,
            Microsoft.Web.Services,
            Version=1.0.0.0,
            Culture=neutral,
            PublicKeyToken=31bf3856ad364e35" />
      </configSections>

      <microsoft.web.services>
            <security>
                  <passwordProvider
                  type="WSE_Security.WSEPasswordProvider,  WSE_Security"
                  />
            </security>
      </microsoft.web.services>

      <system.web>

            <!--  DYNAMIC DEBUG COMPILATION
            Set compilation debug="true" to insert debugging symbols
            (.pdb information)
            into the compiled page. Because this creates a larger file
            that executes
            more slowly, you should set this value to true only when
            debugging and to
            false at all other times. For more information, refer to
            the documentation about
            debugging ASP.NET files.
            -->
            <compilation defaultLanguage="vb" debug="true" />

            <!--  CUSTOM ERROR MESSAGES
            Set customErrors mode="On" or "RemoteOnly" to enable custom
            error messages, "Off" to disable.
            Add <error> tags for each of the errors you want to handle.

            "On" Always display custom (friendly) messages.
            "Off" Always display detailed ASP.NET error information.
            "RemoteOnly"  Display  custom  (friendly)  messages  only  to
            users not running
            on  the  local  Web  server.  This  setting  is  recommended  for
            security purposes, so
            that  you  do  not  display  application  detail  information  to
            remote clients.
            -->
            <customErrors mode="RemoteOnly" />

            <!--  AUTHENTICATION
```

```
This section sets the authentication policies of the
application. Possible modes are "Windows",
"Forms", "Passport" and "None"

"None" No authentication is performed.
"Windows" IIS performs authentication (Basic, Digest, or
Integrated Windows) according to
its settings for the application. Anonymous access must be
disabled in IIS.
"Forms" You provide a custom form (Web page) for users to
enter their credentials, and then
you authenticate them in your application. A user
credential token is stored in a cookie.
"Passport" Authentication is performed via a centralized
authentication service provided
by Microsoft that offers a single logon and core profile
services for member sites.
-->
<authentication mode="Windows" />


<!--  AUTHORIZATION
This section sets the authorization policies of the
application. You can allow or deny access
to application resources by user or role. Wildcards: "*"
mean everyone, "?" means anonymous
(unauthenticated) users.
    -->
<authorization>
      <allow users="*" /> <!-- Allow all users -->

      <!--   <allow       users="[comma separated list of
      users]"
      roles="[comma separated list of roles]"/>
      <deny      users="[comma separated list of users]"
      roles="[comma separated list of roles]"/>
      -->
</authorization>

<!--  APPLICATION-LEVEL TRACE LOGGING
Application-level tracing enables trace log output for
every page within an application.
Set trace enabled="true" to enable application trace
logging.  If pageOutput="true", the
trace information will be displayed at the bottom of each
page.  Otherwise, you can view the
application trace log by browsing the "trace.axd" page from
your web application
root.
-->
<trace enabled="false" requestLimit="10" pageOutput="false"
traceMode="SortByTime" localOnly="true" />


<!--  SESSION STATE SETTINGS
By default ASP.NET uses cookies to identify which requests
belong to a particular session.
```

```
            If cookies are not available, a session can be tracked by
            adding a session identifier to the URL.
            To disable cookies, set sessionState cookieless="true".
            -->
            <sessionState
            mode="InProc"
            stateConnectionString="tcpip=127.0.0.1:42424"
            sqlConnectionString="data
            source=127.0.0.1;Trusted_Connection=yes"
            cookieless="false"
            timeout="20"
            />

            <!--  GLOBALIZATION
            This  section  sets  the  globalization  settings  of  the
            application.
            -->
            <globalization                          requestEncoding="utf-8"
            responseEncoding="utf-8" />

            <webServices>
                  <soapExtensionTypes>
                        <add
                        type="Microsoft.Web.Services.WebServicesExtensi
                        on, Microsoft.Web.Services,
                        Version=1.0.0.0,
                        Culture=neutral,
                        PublicKeyToken=31bf3856ad364e35"    priority="1"
                        group="0" />
                  </soapExtensionTypes>
            </webServices>

      </system.web>

</configuration>
```

Appendix F
Client application's web.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

      <configSections>
            <section                        name="microsoft.web.services"
            type="Microsoft.Web.Services.Configuration.WebServicesConfi
            guration,
            Microsoft.Web.Services,
            Version=1.0.0.0,
            Culture=neutral,
            PublicKeyToken=31bf3856ad364e35" />
      </configSections>

      <microsoft.web.services>
            <security>
                  <decryptionKeyProvider
                  type="WSE_SecurityClient.DecryptionKeyProvider,
                  WSE_SecurityClient" />
            </security>

            <diagnostics>
                  <trace                                    enabled="true"
                  input="C:\Temp\inputTrace.config"
                  output="C:\Temp\outputTrace.config" />
            </diagnostics>
      </microsoft.web.services>

      <system.web>

            <!--  DYNAMIC DEBUG COMPILATION
            Set compilation debug="true" to insert debugging symbols
            (.pdb information)
            into the compiled page. Because this creates a larger file
            that executes
            more slowly, you should set this value to true only when
            debugging and to
            false at all other times. For more information, refer to
            the documentation about
            debugging ASP.NET files.
            -->
            <compilation defaultLanguage="vb" debug="true" />

            <!--  CUSTOM ERROR MESSAGES
            Set customErrors mode="On" or "RemoteOnly" to enable custom
            error messages, "Off" to disable.
            Add <error> tags for each of the errors you want to handle.

            "On" Always display custom (friendly) messages.
            "Off" Always display detailed ASP.NET error information.
            "RemoteOnly"  Display  custom  (friendly)  messages  only  to
            users not running
            on  the  local  Web  server.  This  setting  is  recommended  for
            security purposes, so
```

```
that you do not display application detail information to
remote clients.
-->
<customErrors mode="RemoteOnly" />

<!--  AUTHENTICATION
This section sets the authentication policies of the
application. Possible modes are "Windows",
"Forms", "Passport" and "None"

"None" No authentication is performed.
"Windows" IIS performs authentication (Basic, Digest, or
Integrated Windows) according to
its settings for the application. Anonymous access must be
disabled in IIS.
"Forms" You provide a custom form (Web page) for users to
enter their credentials, and then
you authenticate them in your application. A user
credential token is stored in a cookie.
"Passport" Authentication is performed via a centralized
authentication service provided
by Microsoft that offers a single logon and core profile
services for member sites.
-->
<authentication mode="Windows" />


<!--  AUTHORIZATION
This section sets the authorization policies of the
application. You can allow or deny access
to application resources by user or role. Wildcards: "*"
mean everyone, "?" means anonymous
(unauthenticated) users.
    -->
<authorization>
      <allow users="*" /> <!-- Allow all users -->

      <!--   <allow       users="[comma separated list of
      users]"
      roles="[comma separated list of roles]"/>
      <deny      users="[comma separated list of users]"
      roles="[comma separated list of roles]"/>
      -->
</authorization>

<!--  APPLICATION-LEVEL TRACE LOGGING
Application-level tracing enables trace log output for
every page within an application.
Set trace enabled="true" to enable application trace
logging.  If pageOutput="true", the
trace information will be displayed at the bottom of each
page.  Otherwise, you can view the
application trace log by browsing the "trace.axd" page from
your web application
root.
-->
```

```
            <trace enabled="false" requestLimit="10" pageOutput="false"
            traceMode="SortByTime" localOnly="true" />


            <!--  SESSION STATE SETTINGS
            By default ASP.NET uses cookies to identify which requests
            belong to a particular session.
            If cookies are not available, a session can be tracked by
            adding a session identifier to the URL.
            To disable cookies, set sessionState cookieless="true".
            -->
            <sessionState
            mode="InProc"
            stateConnectionString="tcpip=127.0.0.1:42424"
            sqlConnectionString="data
            source=127.0.0.1;Trusted_Connection=yes"
            cookieless="false"
            timeout="20"
            />

            <!--  GLOBALIZATION
            This   section   sets   the   globalization   settings   of   the
            application.
            -->
            <globalization                          requestEncoding="utf-8"
            responseEncoding="utf-8" />

      </system.web>

</configuration>
```

Appendix G
SOAP request

```
<soap:Envelope    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <soap:Header>
            <wsrp:path
            soap:actor="http://schemas.xmlsoap.org/soap/actor/next"
            soap:mustUnderstand="1"
            xmlns:wsrp="http://schemas.xmlsoap.org/rp">
                  <wsrp:action>http://fishbonesystems.com/AccountServic
                  e/GetCreditAccountDetails</wsrp:action>
                  <wsrp:to>http://segots0002/MasterProject.WS/AccountSe
                  rvice.asmx</wsrp:to>
                  <wsrp:id>uuid:a308a838-28c4-47a9-a11d-
                  bc748317c6b8</wsrp:id>
            </wsrp:path>
            <wsu:Timestamp
            xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
                  <wsu:Created>2003-11-24T15:39:37Z</wsu:Created>
                  <wsu:Expires>2003-11-24T15:44:37Z</wsu:Expires>
            </wsu:Timestamp>
            <wsse:Security                        soap:mustUnderstand="1"
            xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext">
                  <wsse:UsernameToken
                  xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/util
                  ity"    wsu:Id="SecurityToken-655b6849-aa2c-47ce-987f-
                  fe28d53612dc">
                        <wsse:Username>MartinAntonsson</wsse:Username>
                        <wsse:Password
                        Type="wsse:PasswordDigest">Q8LhR5JNa+LAEiuHFq8r
                        Scp6EUI=</wsse:Password>
                        <wsse:Nonce>aNw2dNCg0Vg6G7j1um4W3w==</wsse:Nonc
                        e>
                        <wsu:Created>2003-11-24T15:39:37Z</wsu:Created>
                  </wsse:UsernameToken>
            </wsse:Security>
      </soap:Header>
      <soap:Body>
            <GetCreditAccountDetails
            xmlns="http://fishbonesystems.com/AccountService" />
      </soap:Body>
</soap:Envelope>
```

Appendix H
SOAP response

```
<soap:Envelope    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <soap:Header>
            <wsu:Timestamp
            xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility">
                  <wsu:Created>2003-11-24T15:39:39Z</wsu:Created>
                  <wsu:Expires>2003-11-24T15:44:39Z</wsu:Expires>
            </wsu:Timestamp>
            <wsse:Security                        soap:mustUnderstand="1"
            xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext">
                  <xenc:ReferenceList
                  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                        <xenc:DataReference      URI="#EncryptedContent-
                        0897d147-47f9-4057-9f38-66e0e14588da" />
                  </xenc:ReferenceList>
            </wsse:Security>
      </soap:Header>
      <soap:Body
      xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
      wsu:Id="Id-dbfea7c1-f05f-44bb-a5f5-e0fced024a7e">
            <xenc:EncryptedData      Id="EncryptedContent-0897d147-47f9-
            4057-9f38-66e0e14588da"
            Type="http://www.w3.org/2001/04/xmlenc#Content"
            xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                  <xenc:EncryptionMethodAlgorithm="http://www.w3.org/20
                  01/04/xmlenc#tripledes-cbc"/>
                  <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                        <KeyName>http://fishbonesystems.com/symmetricKe
                  y</KeyName>
                  </KeyInfo>
                  <xenc:CipherData>
                        <xenc:CipherValue>
                              8O2PBksDhE+K2GkhKPWP8cuIyk1xlSS5Mb1XRJqfZ
                              Qf4rTbuqz0MwYSMQbns13k5gXC0/iGvgq02/u/S6a
                              LW9EnLjaV1qmXOC873AhRjjnoTRXGEZINvDFGZ4ER
                              iiaZD6O0sU3IYGGZptqs8pffeK9FoBBVfyuJRDMn0
                              f0QXiWT3sZMVHF22oEtpqH2ni3OUWDmwcRY6usUW5
                              sQCeiJ05jr2s2Jxa90T8LgBIMlQJYbwiPyNDgF9cP
                              36EQOsHj7VY42ugNK7H0iTDwqSRCpWqtaXlBL/PJv
                              L3N72QMOYfsYs8VBr108BeWG8eu+TWfzZuZhz1usZ
                              pVd20QIXWEDuGyrQluL3tVLq2XttYIsd86HatTBPl
                              THMlXAjE5Szu9SBeGwLPvJOFafLePp3LhaKu4v/Qz
                              a9rATuQcj1+DtrVUdFPw2gh5ZVQYbyIqEDWw/Cg+f
                              /2amaSh3tp59yBYrHsoFdGcYUeuGo2xuR+yLO+eVP
                              Zfqtj5zCcOiuOKgKmWZvBMi4fzZ3FDY0B+UY+qDCE
                              5CMqIIKOEPsRnoQKD7GeJRKIOpcBcB6IU7NyusWV7
                              jheYltg+etal2d/2H7QOA8R+4pRyX4hkfTG1hNxwB
                              hWPtGB4bgNxFCUahjorUHQZFrNcfzEGN6OHb4qNa0
                              YGkeBEgdIeAuYx+9/3bbvR1+IEZLQLndy9a+EtBBC
                              qRevknyLQBNfYGAwGE143KmQlR3g8auucSWYeflps
                              b0eUolLaVRClOOLwzhvZCmyqvo/FJWiM/pSOopkvS
                              ZkeHukxXbrtKn0xGukmb6DaHf/H9qcbmfqtLOKyAG
```

```
                                    R2u5P6pj3XuObluAsDOXmlK6OQ3H7bJVhBIgXdpBK
                                    fj8OR9xuPWZ74acQJXHDEaJZzSdOsl31QJMRz8eM2
                                    mDPyLeQqUYYbdtdpKeUIewrAh62apYUolg+d808kZ
                                    v5HQLbfbxc2EI9JDTD9JshUDFUZjaMRqt3RNG6NEJ
                                    LKCoh3M0x+MilDXcPcoVMLldnOU4txu/A+iy5ZH1o
                                    BLGcxaf6Lvy+bnyTGWAvE5BcRxRC2DrBCfejydY/4
                                    /cDyKYUUQrvJ7d8bSENwFSH6uC0xXZjI4/uXwGSQW
                                    OyPcHv36Z0TOyhjy5jfbIuM7GevWBTjetTdo2mivR
                                    S73wj+Lmc2ZdvBnyhMJSZwsO+iE5UszBNUh508ZlE
                                    8J10vXD83W4mp7LbJUT4GSWifbMNGv2wdzTh/o=
                            </xenc:CipherValue>
                        </xenc:CipherData>
                    </xenc:EncryptedData>
                </soap:Body>
            </soap:Envelope>
```

Appendix I
AccountService.aspx.vb

```
Imports System
Imports System.Xml
Imports System.Xml.Xsl
Imports Microsoft.Web.Services.Security
Imports WSE_Security_BLT

Public Class AccountService
    Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()>            Private            Sub
InitializeComponent()

    End Sub

    'NOTE: The following placeholder declaration is required by the Web Form
Designer.
    'Do not delete or move it.
    Private designerPlaceholderDeclaration As System.Object

    Private   Sub   Page_Init(ByVal   sender   As   System.Object,   ByVal   e   As
System.EventArgs) Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()
    End Sub

#End Region

#Region "Designer code"
    'Log in
    Protected WithEvents lblUserName As System.Web.UI.WebControls.Label
    Protected WithEvents lblPassword As System.Web.UI.WebControls.Label
    Protected WithEvents txtUserName As System.Web.UI.WebControls.TextBox
    Protected WithEvents txtPassword As System.Web.UI.WebControls.TextBox
    Protected WithEvents btnLogIn As System.Web.UI.WebControls.Button
    Protected WithEvents lblDescription As System.Web.UI.WebControls.Label
    Protected WithEvents tblLogin As System.Web.UI.WebControls.Table
    'Registration
    Protected WithEvents txtRegUserName As System.Web.UI.WebControls.TextBox
    Protected WithEvents txtRegPassword As System.Web.UI.WebControls.TextBox
    Protected WithEvents txtGivenName As System.Web.UI.WebControls.TextBox
    Protected WithEvents txtSurname As System.Web.UI.WebControls.TextBox
    Protected            WithEvents            btnMakeRegistration            As
System.Web.UI.WebControls.Button
    Protected WithEvents btnRegistration As System.Web.UI.WebControls.Button
    Protected WithEvents tblRegistration As System.Web.UI.WebControls.Table
    'Purchase
    Protected WithEvents txtPurchaseDate As System.Web.UI.WebControls.TextBox
    Protected WithEvents txtStore As System.Web.UI.WebControls.TextBox
    Protected WithEvents txtAmount As System.Web.UI.WebControls.TextBox
    Protected WithEvents tblPurchase As System.Web.UI.WebControls.Table
    Protected WithEvents btnMakePurchase As System.Web.UI.WebControls.Button
    'Log out
    Protected WithEvents btnLogOut As System.Web.UI.WebControls.Button
    'Account details
    Protected WithEvents btnAccountDetails As System.Web.UI.WebControls.Button
```

```
    Protected WithEvents oXmlRecord As System.Web.UI.WebControls.Xml
    'Delete account, records and customer details
    Protected WithEvents tblButtons As System.Web.UI.WebControls.Table
    Protected WithEvents btnDeleteAccount As System.Web.UI.WebControls.Button
    'Label for the date
    Protected WithEvents lblDate As System.Web.UI.WebControls.Label
    'Button to update the account
    Protected WithEvents btnUpdateAccount As System.Web.UI.WebControls.Button
#End Region

#Region "Private Variables"
    'Variable for the UsernameToken that contains the user name and password
    'that are sent in the SOAP-message
    Private oUserNameToken As UsernameToken
    'Variable for the Web Service reference
    Private oWebServiceRef As WSAccountService.AccountService
    Private                        oWebServiceRefHandler                        As
WSAccountServiceHandler.AccountServiceHandler
#End Region

    Private   Sub   Page_Load(ByVal   sender   As   System.Object,   ByVal   e   As
System.EventArgs) Handles MyBase.Load
        'Put user code to initialize the page here
        If Not IsPostBack Then
            Me.tblPurchase.Visible = False
            Me.tblRegistration.Visible = False
            Me.tblButtons.Visible = False
            Me.btnLogOut.Visible = False
            Me.btnUpdateAccount.Visible = False
            Me.lblDescription.Text = "This account service makes it easy for a
customer to log in and control their purchases. It is easy to make a new
purchase and list the account details. Just registrate if you are not a member
and try it out!" + _
                                     "<br><br>The best thing is that your
account details that is returned from the XML Web Service are encrypted which
secures the message's confidentiality!"
            Me.btnLogIn.Attributes.Add("onclick", "return validateLogIn();")
            Dim oDate As DateTime
            oDate = Now()
            Me.lblDate.Text = oDate.ToString("yyyy-MM-dd")
        End If

        Me.btnDeleteAccount.Attributes.Add("onclick", "alert('Ok, you want to
delete your account and records. \nAll unsettled records will be sent to you,
no problem!\nHave fun and take care!');")
    End Sub

    'Log in
    Private  Sub  btnLogIn_Click(ByVal  sender  As  System.Object,  ByVal  e  As
System.EventArgs) Handles btnLogIn.Click
        oUserNameToken      =      New      UsernameToken(Me.txtUserName.Text,
Me.txtPassword.Text, PasswordOption.SendHashed)
        Dim oXmlDocument As XmlDocument
        oWebServiceRef = New WSAccountService.AccountService
        oWebServiceRef.RequestSoapContext.Security.Tokens.Add(oUserNameToken)
        Try
            oXmlDocument = oWebServiceRef.GetCreditAccountDetails
            Dim oTransXsl As New XslTransform
            oTransXsl.Load(Server.MapPath("transformRecord.xsl"))
            oXmlRecord.Document = oXmlDocument
            oXmlRecord.Transform = oTransXsl
            Dim     oCustomer     As     New     Customer(Me.txtUserName.Text,
Me.txtPassword.Text)
```

```
            Me.CustomerID = CType(oCustomer.CustomerID, String)
            Me.UserName = Me.txtUserName.Text
            Me.Password = Me.txtPassword.Text
            Me.tblLogin.Visible = False
            Me.CleanTblLogin()
            oCustomer = Nothing
            Me.CleanTblRegistration()
            Me.tblRegistration.Visible = False
            Me.tblPurchase.Visible = True
            Me.btnLogOut.Visible = True
            Me.tblButtons.Visible = True
            Me.btnUpdateAccount.Visible = True
            Me.btnAccountDetails.Visible = False
            Me.lblDescription.Text = "You are now logged in. Just make your
choices:<br>- If you want to make a purchase, just fill out the form<br><br>-
You can after that get your account details by pressing the Account Details
button(again)" + _
                                     "<br><br>- You can also pay your unsettled
records and get an invoice sent home to you.<br><br>- Or if you want to delete
your account, press the delete button"
        Catch ex As Exception
            Me.lblDescription.Text = "The login failed because of invalid
credientials, please try again!"
        End Try
    End Sub

    'Make the registration visible
    Private Sub btnRegistration_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnRegistration.Click
        Me.CleanTblLogin()
        Me.CleanTblRegistration()
        Me.tblRegistration.Visible = True
        Me.lblDescription.Text = "You have to fill out all the fields to make a
proper registration. Remember your credentials after you have made your
registration."
    End Sub

    'Registration
    Private Sub btnMakeRegistration_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnMakeRegistration.Click
        oWebServiceRefHandler = New
WSAccountServiceHandler.AccountServiceHandler
        Try
            If oWebServiceRefHandler.RegistrateCustomer(Me.txtRegUserName.Text,
Me.txtRegPassword.Text, Me.txtGivenName.Text, Me.txtSurname.Text) Then
                Dim oCustomer As New Customer(Me.txtRegUserName.Text,
Me.txtRegPassword.Text)
                Me.lblDescription.Text = "Congratualations! You are now a
customer with <b>" + oCustomer.CustomerID.ToString + "</b> as Customer id.
Please, try to log in and use the services that offers."
                Me.tblRegistration.Visible = False
            Else
                Me.lblDescription.Text = "Registration failed, try another user
name and/or password!"
                Me.tblRegistration.Visible = True
            End If
        Catch inex As InvalidCastException
            Me.lblDescription.Text = inex.Message
        Catch ex As Exception
            Me.lblDescription.Text = "The service may be temporay not
available."
        End Try
    End Sub
```

```
    'Make a purchase
    Private Sub btnMakePurchase_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnMakePurchase.Click
        oWebServiceRefHandler                          =                          New
WSAccountServiceHandler.AccountServiceHandler
        Try
            If          oWebServiceRefHandler.MakePurchase(CType(Me.CustomerID,
Integer), Me.txtStore.Text, Me.txtPurchaseDate.Text, Me.txtAmount.Text) Then
                Me.lblDescription.Text  =  "The  purchase  succeded.<br><br>The
records are automatically updated and you can now make another purchase by
doing the same procudure once again.<br><br>You can also press Account Details
for returning the details once again."
                Me.GetAccountDetails()
                Me.CleanTblPurchase()
                Me.btnAccountDetails.Visible = True
            Else
                Me.lblDescription.Text = "The purchase failed! Maybe you have
enter an invalid date or not an number in the amount field. Please try again!"
            End If
        Catch icex As InvalidCastException
            Me.lblDescription.Text = "The purchase failed! Maybe you have enter
an invalid date or not an number in the amount field. Please try again!"
        Catch ex As Exception
            Me.lblDescription.Text  =  "The  service  may  be  temporay  not
available."
        End Try
    End Sub

    'Generate the account details
    Private Sub GetAccountDetails()
        oUserNameToken    =    New    UsernameToken(Me.UserName,    Me.Password,
PasswordOption.SendHashed)
        Dim oXmlDocument As XmlDocument
        oWebServiceRef = New WSAccountService.AccountService
        oWebServiceRef.RequestSoapContext.Security.Tokens.Add(oUserNameToken)
        Try
            oXmlDocument = oWebServiceRef.GetCreditAccountDetails
            Dim oTransXsl As New XslTransform
            oTransXsl.Load(Server.MapPath("transformRecord.xsl"))
            oXmlRecord.Document = oXmlDocument
            oXmlRecord.Transform = oTransXsl
            Dim oCustomer As New Customer(Me.UserName, Me.Password)
            Me.CustomerID = CType(oCustomer.CustomerID, String)
            oCustomer = Nothing
        Catch ex As Exception
            Me.lblDescription.Text = "An error occurred updating your records,
please try again later."
        End Try
    End Sub
    'Get account details
    Private Sub btnAccountDetails_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnAccountDetails.Click
        oUserNameToken    =    New    UsernameToken(Me.UserName,    Me.Password,
PasswordOption.SendHashed)
        Dim oXmlDocument As XmlDocument
        oWebServiceRef = New WSAccountService.AccountService
        oWebServiceRef.RequestSoapContext.Security.Tokens.Add(oUserNameToken)
        Try
            oXmlDocument = oWebServiceRef.GetCreditAccountDetails
            Dim oTransXsl As New XslTransform
            oTransXsl.Load(Server.MapPath("transformRecord.xsl"))
            oXmlRecord.Document = oXmlDocument
```

```
            oXmlRecord.Transform = oTransXsl
            Dim oCustomer As New Customer(Me.UserName, Me.Password)
            Me.CustomerID = CType(oCustomer.CustomerID, String)
            oCustomer = Nothing
            Me.lblDescription.Text = "The records that are listed here are all
your purchases. For updating your account (by other means, pay the bill to
Account Service) please press the button 'Pay all unsettled records'!"
        Catch ex As Exception
            Me.lblDescription.Text = "Error occured, please try and log in
again!"
        End Try
    End Sub

    'Delete account, records and customer details
    Private Sub btnDeleteAccount_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnDeleteAccount.Click
        oWebServiceRefHandler = New
WSAccountServiceHandler.AccountServiceHandler
        Try
            If oWebServiceRefHandler.DeleteAccount(CType(Me.CustomerID,
Integer)) Then
                Me.lblDescription.Text = "Your account, records and customer
details are deleted. Thank you for your time here at the Account Service!"
                Me.tblButtons.Visible = False
                Me.CleanTblLogin()
                Me.CleanTblPurchase()
                Me.CleanTblRegistration()
                Me.tblPurchase.Visible = False
                Me.tblRegistration.Visible = False
                Me.tblLogin.Visible = True
            Else
                Me.lblDescription.Text = "Error when deleting."
            End If
        Catch ex As Exception
            Me.lblDescription.Text = "The service may be temporay not
available."
        End Try
    End Sub

    'Update your account, by other means pay your bills
    Private Sub btnUpdateAccount_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnUpdateAccount.Click
        oWebServiceRefHandler = New
WSAccountServiceHandler.AccountServiceHandler
        Dim oXmlDocument As New Xml.XmlDocument
        Try
            oXmlDocument =
oWebServiceRefHandler.UpdateAccountBalance(CType(Me.CustomerID, Integer))
            oXmlDocument.Save("c:/temp/accountBalance.xml")
            Dim oBalance As Integer = 0
            oBalance =
CType(oXmlDocument.GetElementsByTagName("Balance").Item(0).InnerText(),
Integer)
            If oBalance > 0 Then
                Me.lblDescription.Text = "Your account has been updated, by
other means all records upto this date: <b>" +
oXmlDocument.GetElementsByTagName("LastUpdated").Item(0).InnerText + "</b> has
been put on an invoice and been sent to you. The total of the invoice is: <b>"
+ oXmlDocument.GetElementsByTagName("Balance").Item(0).InnerText + "</b>" + _
                                    "<br><br>Are you still that happy?;-)"
            Else
                Me.lblDescription.Text = "You have paid all your records - Good
for you!"
```

```
            End If
        Catch ivex As InvalidCastException
            Me.lblDescription.Text = ivex.Message

        Catch ex As Exception
            Me.lblDescription.Text = "Error when updating the account, please
try again later. The service may be temporary not available."
            Me.lblDescription.Text = ex.Message
        End Try
        Me.GetAccountDetails()
    End Sub

    'Log out
    Private Sub btnLogOut_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnLogOut.Click
        Me.CustomerID = Nothing
        Me.UserName = Nothing
        Me.Password = Nothing
        Me.lblDescription.Text = "You are now logged out!<br>Thanks for your
visit!"
        Me.tblButtons.Visible = False
        Me.CleanTblLogin()
        Me.CleanTblPurchase()
        Me.CleanTblRegistration()
        Me.tblPurchase.Visible = False
        Me.tblRegistration.Visible = False
        Me.tblLogin.Visible = True
    End Sub

    'Clean the fields in the purchase table
    Private Sub CleanTblPurchase()
        Me.txtPurchaseDate.Text = ""
        Me.txtStore.Text = ""
        Me.txtAmount.Text = ""
    End Sub

    'Clean the fields in the registration table
    Private Sub CleanTblRegistration()
        Me.txtRegPassword.Text = ""
        Me.txtRegUserName.Text = ""
        Me.txtGivenName.Text = ""
        Me.txtSurname.Text = ""
    End Sub

    'Clean the fields in the login table
    Private Sub CleanTblLogin()
        Me.txtUserName.Text = ""
        Me.txtPassword.Text = ""
    End Sub

#Region "Private Properties"
    'Property for the customer's customerID
    Private Property CustomerID() As String
        Get
            If IsNothing(Session("CustomerID")) Then
                Return Nothing
            Else
                Return Session("CustomerID")
            End If
        End Get
        Set(ByVal Value As String)
            Session("CustomerID") = Value
        End Set
```

```
    End Property

    'Property for the user name
    Private Property UserName() As String
        Get
            If IsNothing(Session("UserName")) Then
                Return Nothing
            Else
                Return Session("UserName")
            End If
        End Get
        Set(ByVal Value As String)
            Session("UserName") = Value
        End Set
    End Property
    'Property for the password
    Private Property Password() As String
        Get
            If IsNothing(Session("Password")) Then
                Return Nothing
            Else
                Return Session("Password")
            End If
        End Get
        Set(ByVal Value As String)
            Session("Password") = Value
        End Set
    End Property
#End Region

End Class
```

Appendix J
Password provider

```
Imports System
Imports Microsoft.Web.Services.Security
Imports WSE_Security_BLT

Public Class WSEPasswordProvider
    Implements IPasswordProvider

    Private m_sUserPassword As String
    Public Sub New()

    End Sub

    Public Function GetPassword(ByVal UserNameToken As UsernameToken)
    As String Implements IPasswordProvider.GetPassword
        Dim oCustomer As New Customer
        oCustomer.GetCustomerCredentials(UserNameToken.Username)
        Me.UserPassword = oCustomer.Password
        Return oCustomer.Password
    End Function

    Public Property UserPassword() As String
        Get
            Return m_sUserPassword
        End Get
        Set(ByVal Value As String)
            m_sUserPassword = Value
        End Set
    End Property
End Class
```

Appendix K
XML to HTML using XSLT

```
oXmlDocument = oWebServiceRef.GetCreditAccountDetails
Dim oTransXsl As New XslTransform
oTransXsl.Load(Server.MapPath("transformRecord.xsl"))
```

These lines load the transformRecord.xsl and transform the XML to HTML using this XSLT:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html" encoding="UTF-8"/>
       <xsl:template match="RecordList">

       <table border="0" cellpadding="0" cellspacing="0" width="300">
              <tr>
                     <td     align="left"     valign="bottom"     style="padding-
                     left:5px;"><b>Paid</b></td>
                     <td     align="left"     valign="bottom"     style="padding-
                     left:5px;"><b>Purchase Date</b></td>
                     <td     align="left"     valign="bottom"     style="padding-
                     left:5px;"><b>Store</b></td>
                     <td     align="right"     valign="bottom"     style="padding-
                     right:5px;"><b>Amount</b></td>
              </tr>
              <tr>
                     <td colspan="4" valign="top"><hr></hr></td>
              </tr>
              <xsl:for-each select="Record">
              <tr>
                     <xsl:attribute name="bgcolor">
                            <xsl:if test="position() mod 2 = 0">#F7C478</xsl:if>
                     </xsl:attribute>
                     <td align="left" valign="bottom" style="padding-left:5px;">
                            <xsl:variable          name="recordPaid"><xsl:value-of
                            select="Paid"></xsl:value-of></xsl:variable>
                            <xsl:choose>
                                   <xsl:when test="$recordPaid=1">Yes</xsl:when>
                                   <xsl:otherwise>No</xsl:otherwise>
                            </xsl:choose>
                     </td>
                     <td   align="left"   style="padding-left:5px;"><xsl:value-of
                     select="PurchaseDate"/></td>
                     <td   align="left"   style="padding-left:5px;"><xsl:value-of
                     select="Store"/></td>
                     <td  align="right"  style="padding-right:5px;"><xsl:value-of
                     select="Amount"/></td>
              </tr>
              </xsl:for-each>
                     <xsl:choose>
                            <xsl:when test=".//AccountID != '' ">
                                   <tr bgcolor="#CCCCCC">
                                          <td     align="left"     style="padding-
                                          left:5px;"><b>Total</b></td>
                                          <td     align="right"     style="padding-
                                          right:5px;"     colspan="3"><xsl:value-of
                                          select="sum(.//Amount)"></xsl:value-
                                          of></td>
                                   </tr>
                                   <tr bgcolor="#F6BA92">
                                          <td      colspan="4"      align="center"
                                          style="padding-top:5px;padding-
```

```
                                        bottom:5px;padding-left:5px;">Last
                                        updated:<img                    src=""
                                        width="3"/><b><xsl:value-of
                                        select=".//LastUpdated"/></b><img
                                        src=""         width="3"/>(<xsl:value-of
                                        select=".//GivenName"/><img       src=""
                                        width="3"/><xsl:value-of
                                        select=".//SurName"/>)</td>
                                </tr>
                        </xsl:when>
                        <xsl:otherwise>
                                <tr bgcolor="#F6BA92">
                                        <td      colspan="4"      align="center"
                                        style="padding-top:5px;padding-
                                        bottom:5px;padding-left:5px;">You   have
                                        no records in your account</td>
                                </tr>
                        </xsl:otherwise>
                </xsl:choose>
        </table>
    </xsl:template>
</xsl:stylesheet>
```

Appendix L
XML Web Service (AccountService)

```
Imports System
Imports System.Collections
Imports System.ComponentModel
Imports System.Data
Imports System.Diagnostics
Imports System.Web
Imports System.Web.Services
Imports Microsoft.Web.Services
Imports Microsoft.Web.Services.Security
Imports System.Security.Cryptography
Imports System.Security.Cryptography.Xml
Imports System.Xml
Imports System.Web.Services.Protocols

Imports WSE_Security_BLT



<System.Web.Services.WebService(Namespace:="http://fishbonesystems.com/AccountS
ervice")> _
Public Class AccountService
    Inherits System.Web.Services.WebService

#Region " Web Services Designer Generated Code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Web Services Designer.
        InitializeComponent()

        'Add your own initialization code after the InitializeComponent() call

    End Sub

    'Required by the Web Services Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Web Services Designer
    'It can be modified using the Web Services Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()>            Private            Sub
InitializeComponent()
        components = New System.ComponentModel.Container
    End Sub

    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        'CODEGEN: This procedure is required by the Web Services Designer
        'Do not modify it using the code editor.
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

#End Region
```

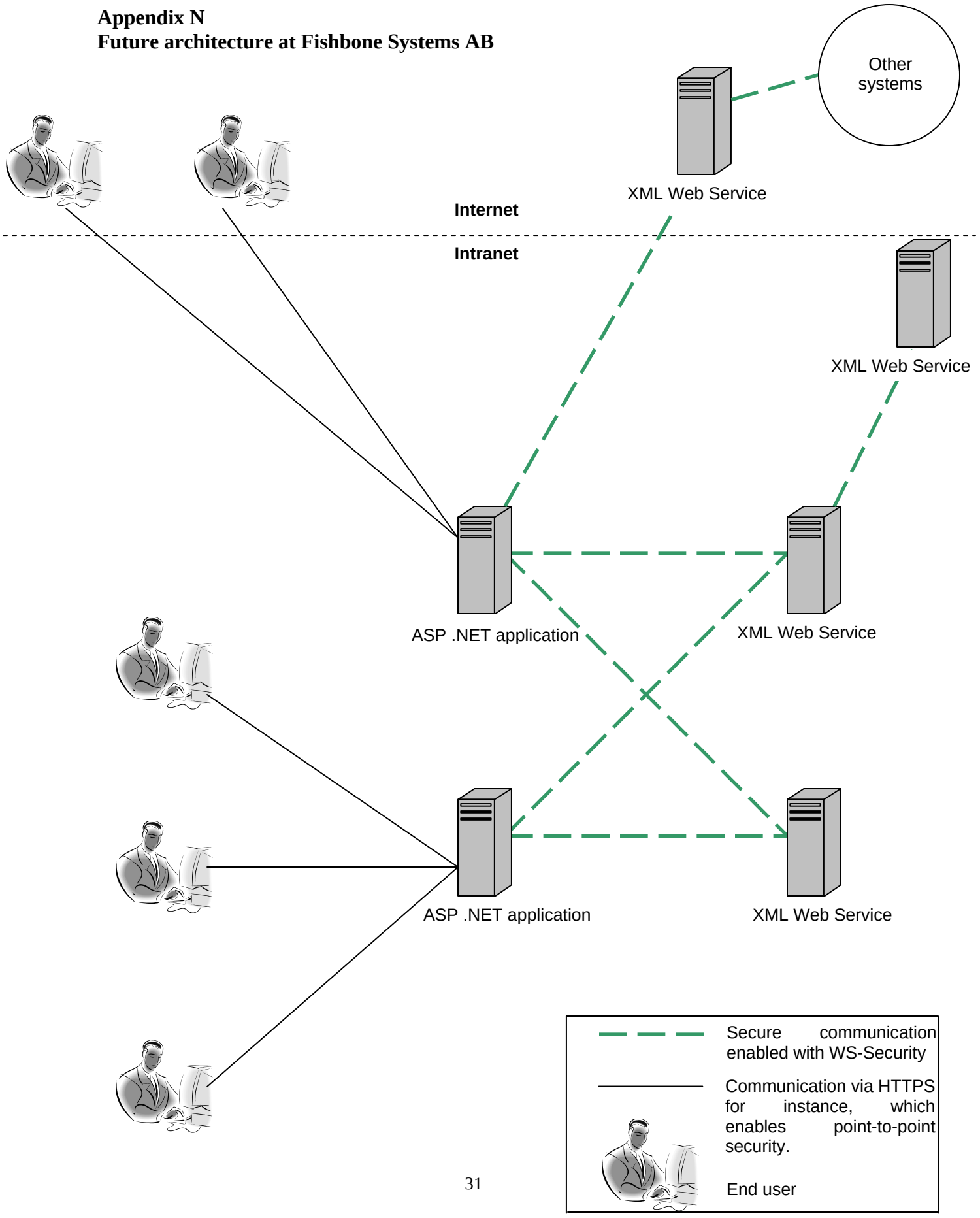
```
#Region "Private Properties"
    Private m_sUserName As String
    Private m_sPassword As String
#End Region

    <WebMethod(MessageName:="GetCreditAccountDetails",        Description:="Gets
details about a credit account. Returns an encrypted XML Document.")> _
    Public Function GetCreditAccountDetails() As XmlDocument
        Dim oXmlDocument As New XmlDocument

        Dim oRequestContext As SoapContext
        oRequestContext = HttpSoapContext.RequestContext
        If IsNothing(oRequestContext) Then
            Throw New ApplicationException("This is not a valid SOAP request!")
        End If

        Dim oUserNameToken As UsernameToken
        oUserNameToken = Me.GetCredentials(oRequestContext)

        If oUserNameToken Is Nothing Then
           Throw  New  ApplicationException("Invalid  Credentials,  Please  try
      again!")
        Else
            Dim oPasswordProvider As New WSEPasswordProvider
            Me.Password = oPasswordProvider.GetPassword(oUserNameToken)
            Me.UserName = oUserNameToken.Username
            Dim oCustomer As Customer
            oCustomer = New Customer(Me.UserName, Me.Password)
            Dim oAccountDetails As New AccountDetails
          oXmlDocument = oAccountDetails.GetAccountDetails(oCustomer.CustomerID)
            Dim oResponseContext As SoapContext
            oResponseContext = HttpSoapContext.ResponseContext

            'This array represens the 128-bit key. Must be identical at the
            client side.
            Dim keyBytes As Byte() = {48, 218, 89, 25, 222, 209, 227, 51, 50,
            168, 146, 188, 250, 166, 5, 206}
            'This array represents the initial vector (IV). Must be identical at
            the client side.
            Dim ivBytes As Byte() = {16, 143, 111, 77, 233, 137, 12, 72}

            Dim    oSymmetricAlgorithm    As    SymmetricAlgorithm    =    New
            TripleDESCryptoServiceProvider

            oSymmetricAlgorithm.Key = keyBytes
            oSymmetricAlgorithm.IV = ivBytes


            Dim oEncryptionKey As EncryptionKey
            oEncryptionKey = New SymmetricEncryptionKey(oSymmetricAlgorithm)

            Dim oKeyName As New KeyInfoName
            oKeyName.Value = "http://fishbonesystems.com/symmetricKey"
            oEncryptionKey.KeyInfo.AddClause(oKeyName)

            Dim      oEncryptedData      As      EncryptedData      =      New
            EncryptedData(oEncryptionKey)
            oResponseContext.Security.Elements.Add(oEncryptedData)

            Return oXmlDocument
        End If
    End Function
```

```
    'Private method that returns a UsernameToken.
    'If it returns nothing, then the user name and password is invalid.
      Private Function GetCredentials(ByVal oRequestContext As SoapContext) As
UsernameToken
        For Each UsrToken As SecurityToken In oRequestContext.Security.Tokens
            If TypeOf UsrToken Is UsernameToken Then
                Return UsrToken
            End If
        Next
        Return Nothing
    End Function
#Region "Private Properties"
    'The user name of the user that is logged in
    Private Property UserName() As String
        Get
            Return m_sUserName
        End Get
        Set(ByVal Value As String)
            m_sUserName = Value
        End Set
    End Property
    'The password that the user enters
    Private Property Password() As String
        Get
            Return m_sPassword
        End Get
        Set(ByVal Value As String)
            m_sPassword = Value
        End Set
    End Property
#End Region
End Class
```

Appendix M
Decryption key provider

```
Imports System
Imports Microsoft.Web.Services
Imports Microsoft.Web.Services.Security
Imports System.Security.Cryptography
Imports System.Security.Cryptography.Xml
Imports System.Xml

Public Class DecryptionKeyProvider
   Implements IDecryptionKeyProvider

    'Constructor
    Public Sub New()

    End Sub


Public Function GetDecryptionKey(ByVal algorithmUri As String, ByVal
keyInfo As KeyInfo) As DecryptionKey Implements
IDecryptionKeyProvider.GetDecryptionKey

      Dim oDecryptionKey As DecryptionKey

      'Recreate the same 16 byte array representing the 128-bit key.
      Dim keyBytes As Byte() = {48, 218, 89, 25, 222, 209, 227, 51, 50,
      168, 146, 188, 250, 166, 5, 206}
      'Recreate the 8 byte (64-bit) array for the initial vector (IV)
      Dim ivBytes As Byte() = {16, 143, 111, 77, 233, 137, 12, 72}

      Dim oSymmetricAlgorithm As SymmetricAlgorithm = New
      TripleDESCryptoServiceProvider
      oSymmetricAlgorithm.Key = keyBytes
      oSymmetricAlgorithm.IV = ivBytes

      oDecryptionKey = New SymmetricDecryptionKey(oSymmetricAlgorithm)

      Return oDecryptionKey
    End Function
End Class
```

Appendix N
Future architecture at Fishbone Systems AB



Future Architecture at Fishbone Systems AB

The end users of Fishbone Systems AB's applications are globally located. This does not mean that the end users only are connected to the applications via the Internet, the communication between user and application could be via an Intranet. The users must be authenticated in some way which enables that only trusted clients communicate with Fishbone Systems AB's systems. There are two scenarios where an end user can access Fishbone Systems AB's web applications, which are illustrated in this Appendix.

- **Intranet**
  A user that is located inside the Intranet communicates with a web application directly via the Intranet. In this scenario, the web application can communicate with XML Web Services that are located inside and/or outside the Intranet.
- **Internet** (Extranet)
  This scenario is similar to this project's implementation. The user is located outside the Intranet. The user has to enter user name and password to login and access the web application. One can refer this to basic authentication, where the user's credentials containing an user name and password is passed in the header of the SOAP request.[24]

Fishbone Systems AB's XML Web Services can be located both inside and outside the Intranet. An XML Web Service can be used by other systems inside the Intranet and if an XML Web Service is located on the Internet, this one can be used by systems that are located outside the Intranet. These XML Web Services could also be used by other companies if Fishbone Systems AB and the calling company have an agreement. This is why Fishbone Systems AB considers the security regarding their XML Web Services as an important issue. The sensitive information that is sent between client applications and XML Web Services must be secured.

The end users can be authenticated if a security token is introduced. Then only authorized clients will be able to communicate with Fishbone Systems AB's systems. The green lines are representing communications where Fishbone Systems AB can guarantee that the message's confidentiality is secured. These goals are fulfilled after this approach is implemented. It assumes that the communication is enabled according to WS-Security as this project's implementation.