Chapter 1: Introduction

Web-services came in the limelight following the need for e-business customers to carry on with business within an organization, and between disparate application environments. The drive was greatly geared by the need to share, combine, and reuse, web-service resources in form of software amongst enterprise organizations. Hence, a growing need to bridge the gap between IT and business became inevitable. Following the evolution of web-services, a number of significant reactions have spun through organizations. According to a market milestone report [DELP02], about 72% of enterprise organizations are making use of web-services largely using the Extended Markup Language (XML) technology. According to the Delphi White Paper [DELP02], of the organizations that had not yet adopted web-services, 27% indicated that experimentation was underway once the benefits of web-services had been made clear to them, through experience-based best practices shared by other organizations. Furthermore, [DELP02] reported that web-service users were afraid to take on the web-service technology since they were concerned about how their security and privacy could be met.

Web-service security was a growing concern amongst web-service clients and users. Since then, a number of web-service standards have been established. The growth in web-service security concern led to the evolution of research into how to enhance security in web-services, particularly the use of Kerberos security to enhance single sign-on (SSO) in a web-service environment. Kerberos is of greater interest since it has proven to offer a SSO feature for the Windows domain system authentication [TUNG99]. Kerberos is an

authentication system used for secure communication and protection of network resources [STAL03, TUNG99]. Consequently there is a need to investigate the applicability of Kerberos technology within the web-service environment for enhancing security. SSO is necessary for web-service users because it gives them the ability to authenticate once for all the protected services required. A number of security standards currently exist to build security into web-services. Some of these include: WS-Security, WS-Trust, WS-Federation, Kerberos Binding, and the Simple Object Access Protocol (SOAP) message security amongst others [ROSE04, NEWC05]. Security components have been integrated into the XML based SOAP messages in order to achieve confidentiality, authentication, authorization, integrity, privacy and availability, which are fundamental components of information security [PLEE03, STEE05]. SOAP is used for communication between web-services; hence there was a need to protect the SOAP messages submitted by the users.

Web-services use the service oriented architecture, with three main activities corresponding to three roles: web-service provider; web-service broker; and the web-service requestor [O'NEI03, ERL04]. Given a scenario where a web-service requestor wishes to access numerous web-service resources located in their domain or across other trust domains, they may require to get authenticated for every web-service resource they are requesting. This is disadvantageous as the web-service requestors' authentication data has to be supplied each time. But, the compromise of this data may result in loss of confidentiality and integrity of the web-service resources in transit and in store. Furthermore, constant submission of authentication data may bring annoyance to the user, especially for compound services.

To address the above problems, we have described how Kerberos security can be used to enhance security in web-services. Kerberos is an authentication architecture used within

organizations as a ticket provider for supporting authentication and session integrity for services requested. Transactions in web-services can now be protected through the use of the application-clients and domain-relay applications both at the web-service environment client-end and server-end. The SOAP requests originating from the client-end (service requestor) are redirected to the application-client who forwards them to the domain-relay. The application-client supplies an identifier of the client together with the ticket for specific service. The domain-relay application (client-end) trusts the application-client to present authentic data belonging to the user. This domain-relay vouches for the user to the domain-relay (server-end) that the information supplied is authentic. Hence, it is the duty of the domain-relay to convince the web-service server that the identification details submitted are indeed authentic. The domain-relays at both the client-end and server-end trust each other to provide correct information. The web-service server has the right to authorize access to web-service resources. The same ticket is used for all services required without supplying the authentication data over and over again. An authenticator (a carrier of client identification details) is generated once for each service requested by the client. It is therefore used to thwart the problem of reuse of stolen tickets. With the guidance of the team at ID-Lab - TNO-ICT, we have together collaboratively developed the architecture for the web-service security.

1.1 Background to the study

Web-services have undertaken the third generation of communication between businesses and customers and involve the interaction between software applications situated at different locations of the web [O'NEI03]. During communication there are threats to confidentiality, integrity, and availability of data. Identification data is normally used for authentication and

authorization to protected network resources. The data in the SOAP messages are protected through XML security, [ERL04, GRAH04, & ROSE04]. SOAP was created to transport XML data from one computer to another over the Hyper Text Transfer Protocol (HTTP). The integrity and confidentiality of SOAP messages is through the use of XML encryption and signature standards. However, for all the SOAP requests, identification data has to be supplied. The concept of SSO evolved because of repetitive authentication required for all the web-services requested. For SSO, Kerberos so far takes the lead due to its current positive performance in the windows operating system.

Kerberos is an authentication system used for secure communication and protection of network resources [STAL03, TUNG99]. Besides, Kerberos has a capability of providing SSO authentication that is why its use was considered for this research. Kerberos being an authentication system ensures that the authentication service is accomplished, paving way for other security services, namely: authorization, confidentiality, data integrity, and auditing and intrusion detection. Kerberos achieves its functionality by the use of tickets to enhance SSO. Users of Kerberos domain are required to acquire tickets used for access to services. All the different services such as File Transfer Protocol (FTP) server, Login server or the Kerberos server itself require different tickets (service granting tickets). For access to web-service resources, the concept of using tickets can be adopted.

Web-services with Kerberos binding have been described in [IBM03b], especially how to integrate Kerberos security into web-service security architecture. However, the focus is on message security, which leaves the gap in the protection of the web-services with Kerberos for the communication level for SSO.

1.2 Problem Statement

Web-services are characterised by the use of the Internet to publish, store, and invoke modular applications over the network. Normally, users (individuals and organizations) have access to numerous organizational resources in a single trust domain by use of Kerberos enabled authentication technology. Kerberos is the prime and securely proven authentication technology used because it is a ticket provider for supporting authentication and session integrity.

In a single trust domain scenario, users wishing to access web-services require to explicitly identify themselves to the web-service providers. Under this circumstance users must present the correct authentication data, which can be accepted by the web-service providers. On the other hand, users may need multiple web-service resources, meaning they will have to be authenticated several times. The need for multiple authentications makes the authentication data susceptible to compromise since it is continuously supplied whenever required. Besides, web-service providers may not trust the web-service requestors well enough to grant them access to resources. Therefore, this shortcoming called for the design of the architecture for enhancing SSO in a web-service environment by use of Kerberos.

1.3 Study objective and scope

The main research goal was to explicitly investigate the feasibility of using Kerberos security for delivering SSO in a web-service environment.

The findings led to the design of the architecture for web-service security.

According to [GOTZ04], SSO is easily tested within a single trust domain rather than in a multiple trust domain. Authentication in a single trust domain is only restricted to one authentication server whereas in a multiple trust domain it involves numerous authentication servers. More significantly, multiple distinct implementations of trust possess authentication models that are mutually incompatible. The challenge on transition from a single to multiple trust domains lies in creating interoperability between them, which creates a complex scenario. This study only focussed on a single trust domain.

This study aimed to achieve the following objective:

- To explore in depth the knowledge about web-service security in relation to single trust domains, multiple trust domains, and Kerberos security.
- To investigate the feasibility of using Kerberos security for SSO authentication within a web-service environment.
- To design an architecture for enhancing SSO in a web-service environment by use of Kerberos.

1.4 Research Questions

This study sought answers to the following research questions that were based on the objectives of the study.

1. To what extent can Kerberos concepts be applied to achieve single sign-on in a web-service environment within a single trust domain?

    a) How is SSO authentication handled in web-services?

b) What is the current authentication challenge faced within a web-service environment? And how can they be resolved?

c) How can the Kerberos concept be applied within a web-service environment to achieve single sign-on authentication?

2. To what extent could Kerberos concepts be applied to achieve single sign-on in a web-service environment within multiple; single trust domains?

a) What is the current trend of Kerberos in relation to web-services within multiple trust domains?

b) Is it possible to adapt Kerberos for single sign-on in a web-service based multiple trust domains?

1.5 Methodology

In order to carry out the research successfully, the following procedure was used. Various literature (textbooks and online articles) were used as a source of knowledge relevant to broaden the researchers knowledge on web-services as well as develop the concept of Kerberos authentication for web-services.

In addition, staff and the participants in the TNO, ID-LAB assisted in guiding and collaboratively developing the requirements for the system. The knowledge gained was used as a guide for designing the architecture for enhancing SSO in a web-service environment by use of Kerberos.

1.6 Relevance

Research indicates that 23% of organizations are processing XML-based information with their business applications [DELP02]. XML has been in the market's collective consciousness far longer than SOAP, UDDI, and WSDL. Software buyers tend to be more comfortable with XML in concept than with the web-service standards, though their understanding of these standards is increasing [DELP02]. Because XML is being well-accepted, Web-services proved to be more than hype and are here to stay, as evidenced by the software vendors adopting and embracing web-services in their products.

Due to the rampant increase in the use of web-services, communication in a web-service environment is of great concern which formed the basis of this study. Therefore, this research might provide a solution to securing web-service communications with Kerberos authentication service, since no such a solution has been presented in the accessed previous studies. At the end of the research, the developed architecture shall be used for enhancing web-service security, within Enterprises and Multi-National Companies for securing web-service transactions through use of Kerberos, which also offers the SSO feature. The SSO feature offers access to numerous web-service applications at single sign on.

1.7 Thesis Outline

This thesis includes five chapters. Chapter 1 provides the background, the statement of the problem, the objectives and scope, the research questions, the methodology, the relevance and the thesis outline.

The second chapter reviews the related literature to the study. The chapter discusses current research literature in relation to web-services, single sign on, and Kerberos authentication concept.

The third chapter gives the system analysis, and outlines the user and server security requirements, used as a basis for designing the architecture.

The fourth chapter provides the architectural design with the description of the message exchanges and authentications in the communication.

The fifth chapter provides the discussion of the findings of the research that address the research questions raised in chapter one. This chapter also draws the conclusions and spells out the limitations of the study. The chapter ends with recommendations and suggests further areas of research that could extend this study.

1.8 Review

The need for achieving SSO through the use Kerberos has been identified. The chapter pointed out the key objective to be addressed in the study. Two research questions were advanced to direct the study. The library search methodology was suggested and the relevance of the study highlighted. The synopsis of the thesis was then spelt out. The next chapter reviews the related literature to the study.

Chapter 2: Review of Related Literature

Web-services are an emerging technology that has been used for application integration as well as universal application connectivity. Web-services require secure communication and message security for its acceptance amongst its users within organizations. Web-service first generation technologies [ROSE04] were created to enable Service Oriented Architecture (SOA) to run in an open Internet. The essence of SOA was to have universal application integration by making an application to act as a service hub while using the web as the middle ware. The first generation web-service technologies are mainly three, namely: WSDL; SOAP, and UDDL. Officially, since SOAP is not simple and has nothing to do with objects and goes beyond access, its acronym is now standalone hence not an acronym anymore. SOAP is used as communication protocol for XML web-services. The WSDL is used for describing web-service message formats such that users can easily know how to structure their SOAP messages directed to specific web-service providers. The UDDI gives a description and location of web-services such that it can easily be used. The security concern in web-services is focused on SOAP, since this is the vehicle to transport web-service requests or response communication. Hence, SOAP becomes one of the core areas of discussion herein.

The growth of web-services led to the growing research in the areas of Single Sign-On for web-services. Different technologies have been described on how SSO can be achieved both for network or Internet and distributed applications as well as web-services. An

overview of the various technologies proposed for achieving SSO is now provided. Directed by the research interest, we present a theoretical analysis of how Kerberos technology has been used and how it can further be applied to carry SSO functionality within a web-service environment is presented. Since web-services will act as founding environment for the study, the understanding of the key concepts of a web-service environment and its security framework are first presented.

2.1 Web Services

Web-services and SOA are often thought to be synonymous. However, a web-service is just an implementation of SOA. SOA was created to provide service through sharing of dynamic application by creating a service provider – service requestor scenario. In the recent development, SOA offers the concept of distributed computing whereby applications call functionality from other applications over networks. Practically, functionality is published on the network or Internet with two capabilities provided: discovery - the ability to find functionality and binding - the ability to connect to the functionality. Two kinds of service users exist: (1) service providers and (2) service requestors. Both make use of the above functionalities. A service provider is responsible for creating a service description and publishing it in an environment where it can be discovered. A service requestor is responsible for finding the service descriptions that have been published in a service registry. A service registry is the location where all the services can be accessed. There are three main roles in SOA, namely: (1) service provider; (2) service requestor; and (3) service registry. The concept of web-services has been built around SOA. The three roles in SOA correspond to three roles in web-services, namely: (1) web-service provider; (2) web-service requestor; and

(3) web-service broker. For web-services, the three roles are accomplished by three different functionalities, namely the WSDL, the UDDI and the SOAP.

WSDL is used to describe web-services. It is used to describe the message syntax associated with the invocation and response of a web-service. In principle, it gives a description of what a web-service does, how a service can be accessed and where it is located. Specifically, it gives details about the data and data formats, the location on the network, in terms of the Universal Resource Locator (URL) of the service. This information is made available by the WSDL schema, which is a description of the web-services. As soon as the web-services have been described, they need to be discovered by the service requestors. The UDDI is a form of service registry. It makes available the service descriptions, previously offered by the service providers, as search results to the service requestors. The services offered by the service providers need to be communicated or delivered to the service requestors.

2.1.1 Description of SOAP

The SOAP protocol takes the role of messages and has predefined framework of how messages should be represented. A typical message carries a SOAP header and body. The SOAP body normally contains the response to the service requested, and if included, the encryption and digital signature. The SOAP header contains security tokens used for protecting specific parts of the SOAP message. In this case, the ability of SOAP to contain security information is of great importance in any upcoming security technologies. Therefore, the SOAP header creates an opportunity to carry more security information required for protecting web-service communication in general.

SOAP defines a communication standard between services using HTTP to carry a simple-like Remote Procedure Calls (RPC) and their results. SOAP messages are based on XML which allows for transfer of information between services in a decentralized and distributed environment. Typically, a SOAP message has one main component, the SOAP envelope which contains two additional components as shown in the Figure 2.1 below and an additional SOAP attachment component. The attachment is normally optional, however it is included when necessary.

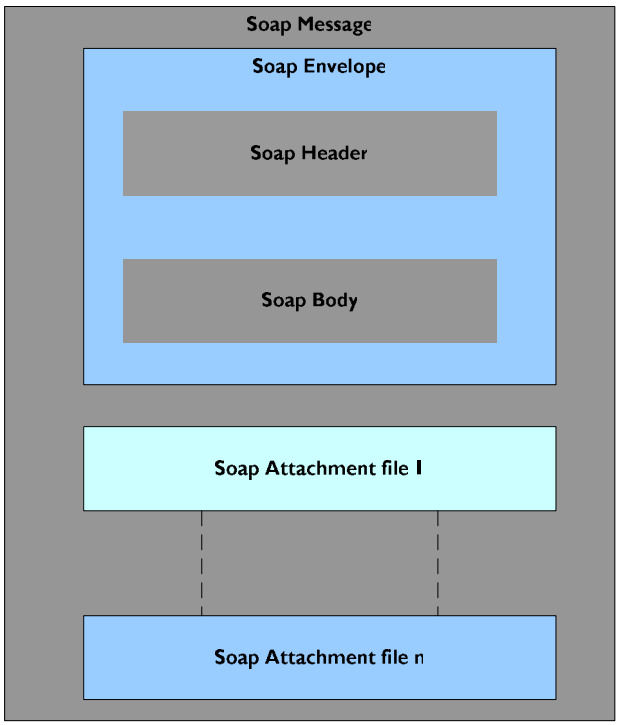

Figure 2.1 Components of a Soap Message

The SOAP envelope is the container for the whole information based on XML. The envelope contains two elements: the SOAP header and the SOAP body. The SOAP header contains information to be passed on to the intermediary before reaching the destination. This could be the URL for locating the service or security information. The SOAP body contains a description of XML data which normally are the SOAP messages. There are three

different types of SOAP messages, namely a request, a response and a fault, that send detailed information about occurrence of an error. A SOAP attachment part (optional) is used when non-XML or coded data needs to be transferred. Therefore, a complete message with an attachment is called a SOAP message package and uses Multipurpose Internet Mail Extensions (MIME), to extend the message by additional body parts that can contain differently coded data [ROSE04, GRAH04, and STEE05].

2.1.2 History of Web Services

In the heyday of Web-services (WS), several challenges and threats interrupted its smooth adoption. According to Rosenberg et al. in [ROSE04], identification and asset protection were great concern at the time of adoption of web-services.

1. The identities of users connecting to company computers/resources were not directly established and that posed a serious threat to company resources. It was difficult to prove who had accessed their resources, since user identities were not available.

2. Asset protection was a major concern since there was a high risk of corporate intellectual property compromise. This is because the company had no means of proving what information that had been accessed and what it would be used for after it left the organizational boundaries. Therefore, the main concern was on how to prove what actions had taken place.

A number of scholars gave leading information on how to resolve the above challenges. According to [ROSE04, GRAH04, ERL04, and STEE05], it was necessary to adopt the use of Secure Assertion Markup Language (SAML). SAML is used for creation and assertion of

identities. It also provides a mechanism for both authentication and authorization processes. Therefore, if SAML is adopted, it enhances web-service security. Similarly, SAML apart, XML Encryption and XML Signature also ensure authentication, authorization, confidentiality, integrity and non-repudiation of web-services both in transit and in storage. Subsequently, SAML assertions are normally included in the SOAP headers and partly in the SOAP body.

Most of the time, when processes cross application boundaries information needs to be shared. For web-services, secure communication is managed through SAML, XML security and WS-Security. However, as the level of inter and intra web-service communication grows, web-service clients are faced with the challenge of re-authentication for all the web-services they wish to access, leading to the need for SSO. According to [ROSE04, Liberty Alliance Project has established a mechanism to use SAML authentication assertion across multiple security domains.

2.1.3 Introduction to Web Service (WS) Technologies

The second generation web-service technologies were introduced because of the need to enhance web-service security. In the first generation technologies, the basic form of security was in XML encryption and XML signature. There was therefore need to have a web-service security to cover the whole web-service framework. This need led to the evolution of the second generation web-service technologies. The new technologies did not render the first generation technologies useless. However, their introduction was meant be used with the first generation technologies, since the first generation technologies form the basis of the functioning of web-service. These technologies include: WS-Coordination, WS-Transaction, WS-Security, WS-Reliable Messaging, BPEL4WS, WS-Policy, WS-Attachments, and WS-

Addressing. In addition, for the establishment of trust and use of policies, the WS-Security framework was developed to resolve the upcoming concerns of secure communication between web-services. All these technologies use SOAP for communication and SOAP is built based on XML.

2.1.4 Security Framework

On top of the SOAP foundation, the WS-Security framework comprises[1] the following standards: WS-policy, WS-Privacy, WS-Trust, WS-Federation, WS-Authorization, and WS-Secure Conversation, which all build on the basics of the WS-Security standard. WS-Security is just one specification. It does not provide a complete security solution to web-services. It needs other standards to perform all the security services for secure web-services. Hence, it co-operates with WS-Trust, WS-Federation, WS-Privacy and WS-Policy. These standards [GRAH04] are described in the next subsections. These standards are shown in Figure 2.2.

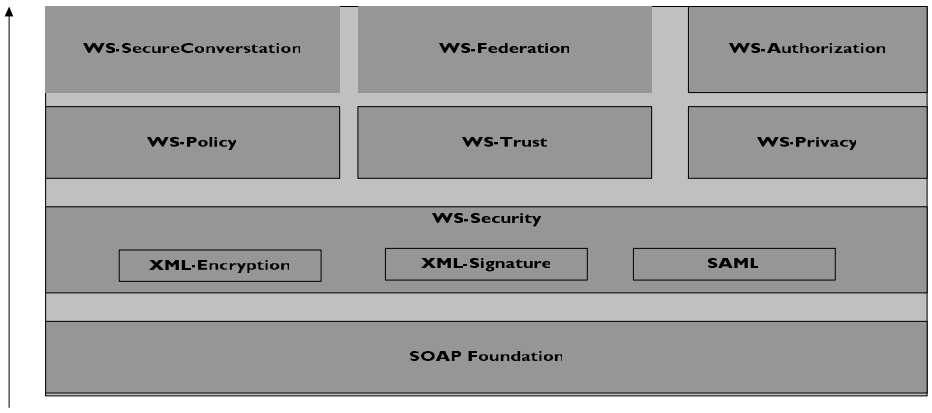

Figure 2.2 WS-Security Framework

[1] Since web-services and especially its security framework is evolving, this is only a current view.

WS-Security

WS-Security provides enhancements to SOAP messaging security through message integrity and message confidentiality. Message integrity is achieved by use of XML Signature while message confidentiality is achieved by use of XML Encryption. WS-Security also provides a mechanism for associating security tokens and key technologies with the messages. Security tokens are used for authentication and authorization.

Security Tokens: The security token may include: username and password, X.509 certificate, SAML assertions and Kerberos tickets. During XML encryption and XML signature, a security token is used within the SOAP header or the body itself depending on the specifications. In this context, Kerberos tickets have been used as binary security tokens as in [ROSE04, GRAH04, ERL04, and STEE05]. In [STEE05], the main challenge lies in the way in which organizations manage different services from different trust domains due to constant multiple sign on required. Therefore, multiple sign on is a big issue that needs to be resolved. According to [ROSE04], Kerberos does not play a leading role in web-service security; because the Kerberos tickets are only used as security tokens built into SOAP headers, as binary security tokens.

Key Technologies: The security tokens are based on shared key technologies, such as shared key encryption and Kerberos. These maintain the use of a shared key between communicating parties. The key is transmitted over the same communication channel by use of public key encryption to ensure confidentiality of the shared key. This technology exhibit a number of advantages over the public key technologies that include: (1) It is much faster than public key encryption; and (2) it can operate on arbitrary large plaintext messages whereas public key encryption cannot and it utilises public key encryption to manage

distribution of the shared key securely to the recipient. This technology has a problem of distribution of the keys. The communicating parties need to get the shared secret key over a secure channel and this is only possible through public key encryption.

Public key technologies have got some advantages over secret key technologies. Two main public key technologies are: public key encryption and digital signatures. They can deliver integrity, non-repudiation, and authentication to XML messages transported using SOAP. However, they have some limitations such as; the key generation is based on two large multi-hundred bit prime numbers that are near the same length, hence making them much slower than the secret key algorithms. It is for this reason that secret key encryption and Kerberos tickets are used as security tokens.

Ws-Policy

WS-Policy describes how service requestors and service providers specify their requirements and capabilities. These could be in terms of: what security token was used; which encryption mechanism should be used; what part of the message must be encrypted and signed; and in terms of SOAP body (part or whole) and the client or server specifications. For the purpose of this research, this policy is limited to message security establishing a communication security policy. The same concept is extended to cover web-service authentication, to allow for SSO. Generally, the policy is an extension to WSDL to describe additional requirements and specifications of the service.

WS-Trust

WS-Trust describes the model for establishing both direct and brokered relationships (including third parties and intermediaries). WS-Trust provides a mechanism for issuing,

renewing and validating security tokens. Trusts can be brokered through creation of security token issuance authorities.

WS-Federation

WS-Federation describes how to construct federated trust scenarios using the WS-Security. The WS-Policy, WS-Trust and WS-Secure Conversation specifications for instance describes how to federate Kerberos and PKI infrastructures.

WS-Privacy

Organizations creating, managing, and using Web-services often need to state their privacy policies and require that incoming requests make claims about the senders' adherence to these policies. A privacy policy might be a specification on the security requirements that must be met by the service requestors before a service can be accessed. In so doing, service requestors have the guarantee that the services offered are secure and the communication is protected. Organizations communicate privacy policies and ensure that web-service requestors comply with these policies. WS-Privacy works in conjunction with WS-Trust and WS-Policy.

2.2 Single Sign-On

[CLER02] defines single sign-on as the ability for a user to authenticate once to a single authentication authority and then gain access to other protected resources without re-authentication. Similarly, single sign-on is a means of sharing authentication data [KHAN03]. The shared data allows for user access to a number of resources on one computer or

different computers (servers) by use of a single authentication, without necessarily re-entering the password [GOTZ04].

Within an organizational setting, there is always a computer network of various resources such as files, Email, and printer servers among others, which usually need to be accessed by the users. Each service has its own idea of a user and how he or she should identify him or her self. This leads to a lot of data being collected for each service required. For a normal user this is not practical since they have insufficient ability to remember all their usernames and passwords, which prompts them to keep recording these details on a piece of a paper. This is a very serious threat to the user in case an impersonator got access to the authentication data. Normally, users experience a great burden of having to sign-on for every service they are accessing, also prompting them to maintain a single username and password for all the services. This is also a serious security threat to the user. The above scenarios led to the evolution of the SSO idea.

According to [PASH03, CLER05], there are a number of reasons for adopting single sign-on. Some of these include: ease of administration, ease of use, and it enables enforcement of coherent security policy. Furthermore, Liberty alliance and Microsoft Passport have had an initiative for SSO. Their aim was to avoid the need for repetitive submission of authentication data by storing demographic information.

Already, the notion of SSO is being used to allow for single access to network resources with single authentication. This is now possible since the latest models of operating systems used in the network environment do have single sign-on capability already incorporated [BEAW05]. In this scenario, identification is performed by a single entity to enable authentication of users for the specific resources. Therefore, a single sign-on environment is

an authentication infrastructure that provides a single authentication authority for all authentication processes by outsourcing authentication processes to a specialized entity that handles authentication for all applications. Participating enterprise applications are required to trust the central authentication authority for the identity of the users. Then, in enterprise applications, users wishing to access multiple applications located across different enterprise networks apply SSO to enable easy access to these applications. The cross-platform authentication is done through use of the Kerberos services for mapping the user accounts to the services that are requested and for which they have authorized access.

The following sections provide further detail on the SSO architecture and technologies, the federation of trust for inter-domain SSO and a justification for our choice of the SSO architecture to be used in a web-service environment.

2.2.1 SSO Architecture and Technologies

There are various technologies that have been created to resolve the single sign on problem. SSO may be achieved through three main approaches: (1) client-based SSO; (2) server-based SSO; and (3) service based SSO [GOTZ04]. Recently a fourth category was suggested by [PASH03] which is the mobile-based SSO.

The service-based SSO has proved its usability and adaptability based on its ability to adapt to the SSO environment with minimal resistance. Furthermore, the service-based SSO contains two approaches thus; the token-based SSO and the Public Key Infrastructure (PKI) based SSO.

Terminology used in SSO

For our description, the common terminology used in the authentication process includes: authentication infrastructure, authentication servers, and authentication authorities. An authentication infrastructure is a set of authentication servers and authentication authorities that provide the authentication service. An authentication server is the physical machine that performs authentication caching the identification credentials of the users. An authentication authority is a trusted logical control of the resources and machines and objects in a domain. These resources include all kinds of servers and the physical machines in the domain.

Client-based SSO (Secure Client-based Credential Caching)

The client-based SSO is a method for password management whereby user credentials or information which the user has to constantly remember is stored and provided each time it is needed. If the credentials are valid the user will be logged on transparently to the resources on the servers. The credentials can be stored on a local machine or in the network domain. For the latter case, there is a need to dedicate a separate domain to control the credential management service. In so doing, the SSO service availability is guaranteed.

In the client-based SSO, cached credentials are used for authentication to the secondary authentication authority, which poses a serious threat to the security of this approach. The application being accessed contacts the local machine directly for authentication data, which is inappropriate, especially if the "bad guys" are within-reach of the authentication credentials. Or if the user would like to access an application globally, then it is not flexible since his credentials are cached locally. Furthermore, if the cached credentials are used, and

a break in the link between the primary credentials and the secondary credentials occurred, then the illegitimate users could manipulate that weakness in the system, which poses a serious threat to the security of the client-based SSO.

Server-based SSO (Secure Server-based Credential Caching)

In the server-based SSO, user passwords are stored in a central repository on a server that provides the needed information directly to the application requesting it. Two kinds of credentials are maintained namely; the primary credentials and the secondary credentials. In this architecture, the primary credential database contains the credentials supplied by the users and also the mappings between a user's primary and secondary credentials. While on the other hand, the secondary credential database stores the user credentials in a database which are used for transparent authentication of the users through synchronization of the primary credentials to the secondary credentials. This means, synchronization can be performed by using one of these three approaches [GOTZ04], namely:

1. The primary credential database includes the synchronization services;

2. An external software handles the credential synchronization or,

3. An administrator or user (self administration) performs the synchronization.

For approaches 1 and 2 named above to take effect, a trust relationship must have been established between the authentication authorities, while for approach 3 it's not necessary. Since the server-based SSO does not outsource the synchronization service due to lack of a trust relationship, then with the administrator in full control of synchronization, there is high risk of erroneous synchronization which may cause compromise to the system.

Service-based SSO

In a service-oriented architecture the management of authentication information has to be provided as a service. From the user's point of view, one authentication gives transparent access to all services that use this single sign-on service [HILL05 & GOTZ04]. Usually a description of the service based architecture is given where SSO is either token-based or PKI as follows:

1. In a token-based SSO environment, the user receives a temporary token after the primary sign-on. This temporary identification is used as proof of authenticity signing on to other authentication authorities that trust the primary authentication authority in order to grant access to their services. Kerberos tickets can be used as a security token within service-based SSO.

2. In a PKI-based SSO environment, the public or private key pair certificates the user's identity to the service provider, based upon which it decides to grant or reject access to its service. Since a trust relationship has been established, any secondary authentication authority will accept such a token or a PKI key pair. This implies that each service provider is to support token-based or PKI-based security in order to implement service-based SSO.

The service-based SSO is secure since the security of the whole environment therefore depends on the authentication authority security standards. Consequently, the improvement in the security level of the authentication authority will lead to the improvement of the security of the whole system.

Mobile-based SSO

An alternative technology used to enhance SSO is Global System for Mobile Communication (GSM) or Universal Mobile Telephone System (UMTS) for SSO, as proposed by [PASH03]. The GSM or UMTS operators are used to authenticate users like an authentication service provider (ASP). In this system, a shared secret is maintained between the Service Provider (SP) and the ASP. The aim of this alliance is to make sure that process to the user, such that they can also supply their authentication information only once for all the services that they wish to gain access. For example, for a user accessing a service, the SP requires authentication assertions from the ASP, which is a notification of the user authentication status. The SP evaluates the authentication assertions to determine whether or not to grant access to a protected resource to the specified user. In this system, the user authentication information is handled by the ASP and the desired SP in such a way that it is transparent to the user.

The idea behind this concept is that a lot of the ASP functionality that is already available at telecom operators to support their GSM or UMTS authorisation, can be reused by other service providers. Since the GSM or UMTS operator now plays the role of the ASP, this concept requires the availability of the mobile telephony networks. If this concept is to be applied for web-services, then we would propose that both GSM or UTMS and other SSO technologies be deployed to cater for disparate network resource locations. For the benefit of this research, we wish to adapt some authentication process concepts for developing an SSO framework for web-service single sign-on with Kerberos technology.

In [MAcD05], a description of a protocol of how GSM or UTMS can be used to secure web-services is given. The proposed protocol involves three main actors, being the

consumer, the Mobile Operator and the service provider. The mobile operator is usually a trusted third party meant to offer SSO authentication services. To this end the mobile operator runs a token distribution center which is used in case a mobile station or consumer needs to access a specific web-service. The mobile operator sends the authentication token directly to the web-service provider. A security agent is used to ensure secure communication and transmission of messages between the mobile operator and the web-service provider.

However, with this scheme, there is a high risk of theft of Subscriber Identity Module (SIM) cards, as this are the means by which consumers can communicate with the web-service providers through the mobile devices. This may lead to impersonation, whereby the consumers' sensitive information may be used for fraudulent activities. This is because the mobile operator scheme gives a single authentication of the consumer to all the services they require. We wish to use the Kerberos security for SSO, relying on the Kerberos ticket to be stored in a central authentication server, instead of on a mobile device whose security could be compromised at any time.

Federation of Trust for SSO

Single sign-on is normally used within a single trust domain but can be extended for use within multiple trust domains. To that end, a federation of trust is to be established between two authentication authorities to solve the problem of different sign-on identities across different trust domains. We use agreements, standards, and technologies to make identity and entitlements portable across autonomous identity domains.

In [HILL05], the SSO framework is presented for web-service-based distributed applications. They use federation of trust for authentication in order to enhance SSO for distributed applications within a web-service environment. Their protocol design is based upon the Kerberos protocol authentication system. In addition, the SSO federation can be extended to the following: Kerberos-based federation, PKI-based federation and SAML-based federation, [CLER02]. The three categories of federation support various functionalities, among which are support for entity authentication, support for data authentication, authorization federation support, granularity of trust relationship and security policy support.

The main advantage of federated SSO environment is the extension of single sign-on beyond the scope of the primary authentication authority of the home domain. A federation of trust allows the user to utilize not only services in his home domain but it also allows the user to utilize services in foreign domains without having credentials for them. However, the federation of trust extends the "the key to the kingdom" problem of single sign-on. The solution to this is establishment of strong security stronghold, such that an intruder will not have access to resources within a single trust domain, as well as across other domains.

[HILL05] states that web-service in organizations currently are protected by web-service firewalls. The firewalls address the issue of access that is distinguishing proper access and improper access but denies not allowed or disallowed clients. Therefore, the issue of identification and authentication to web-service resources is a major concern. An authentication technology has been proposed that is much like the Kerberos authentication architecture. The concept of Kerberos authentication is not used since the claim is that Kerberos authentication is not suitable for highly dynamic networks, and therefore cannot be applied for authentication and authorization of web-service applications. Kerberos

version 4 uses direct trust between authentication domains while Kerberos version 5 is used to improve inter-domain communication but with a static hierarchy of authentication domains. A user still needs to get a security token from his parent authentication domain before gaining access to another domain. Therefore the complexity problem is the main challenge observed.

In our research, the complexity problem is also of concern, however we resolved it by basing the design architecture of the web-service authentication on a single trust domain. In this case we have a single authentication authority and all web-services are registered in the database. In chapter three (Architectural design) of this report, we discuss in detail how Kerberos has been used for building an authentication system for web-services in a single trust domain.

2.2.2 Service-based SSO for Web Service Security.

Given that the client-based SSO is based on a localised credential domain which is susceptible to database attack due to its centralised nature makes it not suitable for use for web-service security. In addition for the server-based SSO, the synchronization of primary and secondary credentials is not outsourced to a trusted synchronization server, leaving the administrator to perform the synchronization function. If this is adopted for use in a web-service environment, it would still leave the security critical web-services in a vulnerable state, since the administrator can easily perform erroneous synchronization. For the service-based SSO, there is an advantage of outsourcing different roles to different servers creating SOA, which is a backbone of web-services. We recommend use of this architecture for defining use of Kerberos security within a web-service environment. We describe the Kerberos security concept in the next section.

2.3 Kerberos Security

Kerberos is a network authentication service that is used for verifying the identities of Kerberos principals e.g. (a workstation user or a network server) located on an open (unprotected) network [RFC4120 and RFC1510]. The Kerberos authentication server plays a number of roles like providing tickets for requested services and establishing secure connections for exchange of the session keys and tickets among others. Privacy and integrity of the message exchanges between the principals can be secured by encrypting the data to be passed by using the session key contained in the ticket or the sub-session key found in the authenticator. In [RFC4537], a mechanism was introduced that allows the clients and servers to negotiate different encryption systems. If a client prefers a given encryption type besides that used in the service ticket, then the client sends a list of supported encryption types in order of preference to the server, who also checks whether the encryption type suggested by the client is available in the server system. A choice is then made of which encryption type to be used over that in the Kerberos encryption system. This is an additional feature in the [RFC4537].

Kerberos authentication service has been proven to offer authentication security within network systems due to the strong encryption mechanisms. Besides, the Kerberos network authentication service provides the SSO functionality [TUNG99]. Our main concern is how Kerberos authentication service can be adopted to deliver SSO within a web-service environment.

The subsequent sub-sections present the key concepts of Kerberos operation, current state-of-the-art initiatives for application of Kerberos in web-service environments and finally a description of the operation of Kerberos.

2.3.1 Key concepts of Kerberos

Kerberos is based on the idea of tickets. A ticket is a data structure that wraps cryptographic keys along with other bits of information to allow users gain access to domain resources. A Key Distribution Center (KDC) distributes Kerberos tickets to authenticated users. A KDC issues two types of tickets: a master ticket also known as a Ticket Granting Ticket (TGT) and a service ticket. A master ticket is issued once the user logs onto the system and it is used for obtaining the service ticket. The service ticket is used by the user to gain access to different servers from which services are required. These tickets can be used for communication within a single domain or realm or for inter-domain communication. A domain or realm is managed by an authentication authority.

In a case where inter-domain communication is involved the authentication authorities from each domain have to establish trust with each other to allow users from each domain to gain services from the other domain. A Kerberos environment consists of a Kerberos server, a number of clients and a number of application servers. The Kerberos server maintains a database of usernames and hashed passwords of the users. All servers are registered in the Kerberos server and a shared secret is maintained between the Kerberos server and the other servers.

In typical Kerberos, Authentication Dialogue attributes include: client, identifier of client, authentication server, ticket granting server, password of user client, network address, encryption, secret key, authenticator, timestamp, lifetime, ticket granting ticket, service ticket, service server, listed below as described in [STAL03].

1. c = Client
2. ID_c = Identifier of user on c,
3. AS = Authentication server,
4. TGS = Ticket Granting Server,
5. P_c = password of user on c,
6. AD_c = Network address of c,
7. E = Encryption
8. K =Secret key shared between principals
9. $Authenticator_c$ = Generated by the client to validate the ticket.
10. TS = Timestamp
11. Lifetime
12. $Ticket_{tgs}$ = Ticket Granting Ticket
13. $Ticket_v$ = Service ticket
14. V= Service Server
15. ‖ = concatenation

Table 2.1 Kerberos authentication dialogue attributes

2.3.2 Description of Kerberos Process of Operation

During initial log on to the domain, the user obtains the master ticket by contacting the AS which is a transparent action to the user. A typical Kerberos authentication flow shown in Figure 2.3 and the process is described as follows [STAL03, STEE05]:

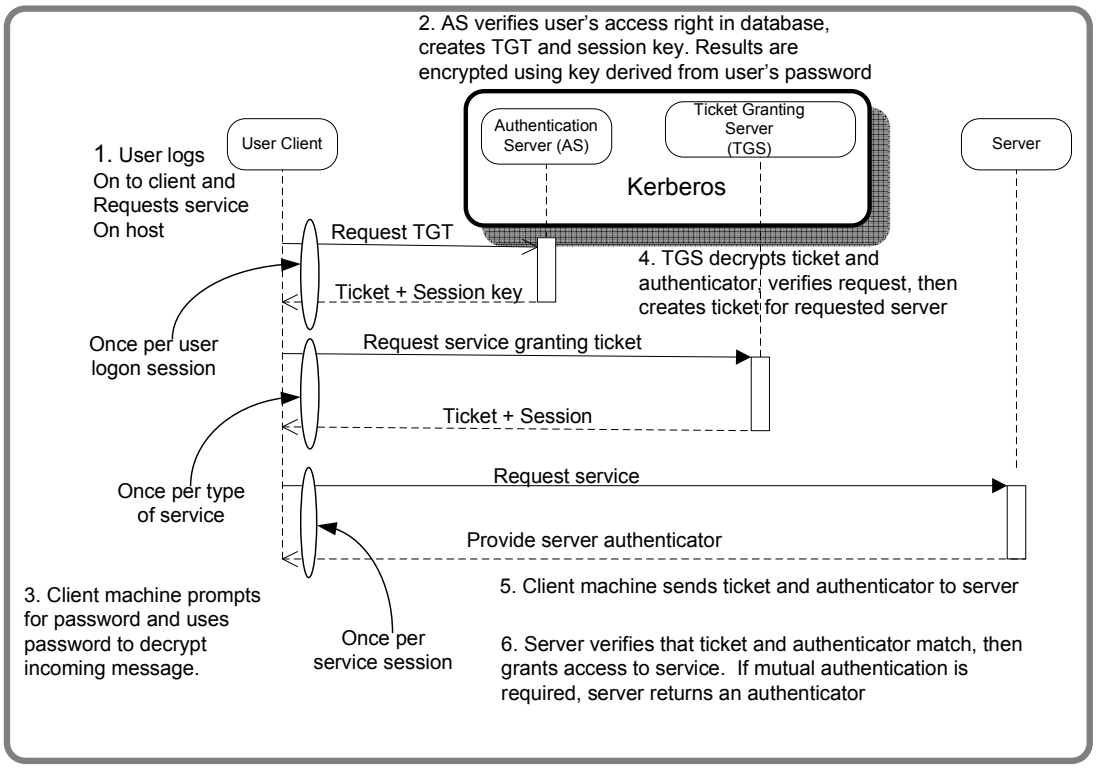

Figure 2.3 Sequence flow in Kerberos authentication

1. The client sends a request to the AS containing the user ID and TGS ID for which services are required once he logs on to his client.

2. The AS verifies the user's access right in the database, and then creates a ticket granting ticket and session key. Both the TGT and session key are encrypted using a key derived from the user's password. The TGT contains the credentials of the TGS and IDc, ADc, lifetime and timestamp of the ticket.

3. The client prompts for the user password, which is used to decrypt the incoming message. After this, the ticket and authenticator are sent to the TGS. The authenticator contains the user's name, network address, and Timestamp.

4. The TGS decrypts the ticket and authenticator and verifies the request, then creates a service ticket for requested server.

5. The client sends the service ticket and authenticator to the server..

The server verifies that the ticket and the authenticator match, and then grants access to the service requested.

2.3.3 State-of-the-art in Kerberos-like authentication services

Kerberos has been used as a low level security mechanism working under General Security Services (GSS) to achieve SSO. In [IBM03b and KHAN03], a description of how to integrate Kerberos security environment with the web-service security architecture is given. According to this report, for Kerberos to be integrated into web-services security, it requires the following aspects; requesting and issuing of security tokens, attaching security tokens to messages, establishing a secure context and signing and encrypting the message using the security context. The GSS-API uses the Kerberos token for signing and encrypting messages, together with the WS-Trust. So, the client applications making use of Kerberos tokens contact the Kerberos KDC directly for request of a security token. Unfortunately, this does not concur with our intended use of Kerberos for SSO since Kerberos here is only deployed as a mechanism for generating security tokens; it does so in such a way that the client applications have to get in contact with the KDC(s) directly. This is a limitation that we intend to resolve by investigating further applicability of Kerberos security for SSO within the web-service environment.

According to [SURF05], A-Select was introduced as an authentication system in institutions of higher learning within the Netherlands. Just like A-Select, Athens is a counterpart of A-

Select being used for authentication in the British institutions of higher learning. In relation to SSO, A-Select authentication service was launched to allow for several applications to make use of a single authentication method (e.g., the same username and password, use of one time codes sent the user's mobile phones, among others) could be used to access several applications [SURF05]. In so doing, SSO is supported where by a user can access several applications after a single log in. The A-Select infrastructure makes use of the A-Select server configures the various authentication methods through as modules, and it sets cookies to be used, such that the user requests are redirected to the cookies each time a previously accessed application is needed. Different applications are added to the A-Select server which can be accessed by the user once per single login.

Therefore, the A-select infrastructure enforces SSO through the A-Select central authentication server. The concept of SSO is broadly catered for in the web-service environment through use of Kerberos authentication service.

Given the literature above, it is evident that Kerberos is not yet established as a mechanism for enhancing SSO within the web-service environment. The only attempt was from Liberty Alliance Project to establish federated identities through SAML, and in the long run establish federated SSO for a web-service environment; however, this is still underway. As a result, we developed a design of the architecture of securing web-services through use of Kerberos authentication service.

2.3.4 Kerberos tickets as Security Assertions in Web Service Enhancements (WES) Specification

As stated earlier, Kerberos has been used as a security token for delivering security for both SOAP and XML. Using WSE 2.0—Web-service Enhancements [2], Web-service communication can be signed and encrypted using Kerberos tickets, X.509 certificates, username and password credentials, and other custom binary and XML-based security token [O'NEI03]. However, WSE is a simplification in the development and deployment of the Web-service Security Framework as presented in section 2.1.4. Developers and administrators can more easily apply security policies on Web-services running on Microsoft's .NET Framework. In addition, an enhanced security model provides a policy-driven foundation for securing Web-services across trust domains. WSE also supports the ability to establish a trust-issuing service for retrieval and validation of security tokens, as well as the ability to establish more efficient long-running secure communication via secure conversations. Security assertions in WSE provide for integrity, confidentiality, and client and service authentication. It is argued that, since WSE is specific to windows platform, any WSE-implementation of Kerberos can interoperate with all web-services belonging to that environment.

[2] This is nothing but a .NET class Library for building or applying the WSE Specifications to baseline Web-services. The WSE specifications include WS-Security, WS-Trust, WS-Policy, WS-Secure Conversation, WS Addressing, WS Messaging, WS-Attachments

2.4 Review

This chapter described in detail the web-services and Kerberos services. The chapter indicated areas of interest from other research works which served as a foundation for the rest of this study. The next chapter gives the analysis of the current web-service system, and sites the problems that need to be addressed.

Chapter 3: System Analysis

We analyze the current use of web-services and identify the current vulnerabilities and security considerations that need to be addressed. We take a case scenario of a user in an organization network like TNO-ICT IDLab. This case creates the platform for describing the need to have an improvement in the current web-service infrastructure to ensure security with SSO enabled.

3.1 Case: A User within the TNO-ICT IDLab

Take an employee of TNO-ICT called Nikita. She is a professional system designer and architect, and sometimes takes on international assignments, which means more international travels. Travelling is one of her special hobbies besides her deep enthusiasm for her job assignments. Therefore, whenever she has a vacation, she takes a holiday in a location that may be geographical different in each holiday occasion. The weather condition always determines her decisions on where to take such holiday trips. This means that she must always keep checking for weather updates prior to her travels.

But, for her to check the weather forecasts she basically has two options:
First, she could use any free weather forecast website information to check for the latest weather forecast update. In this case, she is not obliged to supply any of her personal information or even get authenticated before using the service. The only problem with this kind of weather forecast is that the correctness of the weather data may not be guaranteed because the weather forecast service provider is not under any circumstance obliged to

ensure correctness of its data. In that case, Nikita's need for correct data may not be well addressed. Therefore, since Nikita crucially needs correct data and reliable services, she can better take the second option.

Second, she could register for a paid-for or free weather forecast web-service whereby she either receives weather updates through electronic mail box, or to her mobile phone, or directly through the weather forecast web-service website. In this situation, Nikita's requests and responses for weather data are encapsulated in the SOAP messages with XML, and sent through the SOAP protocol. If a web-service is registered, Nikita is in position to prove the identity of the web-service provider through mutual authentication through the SSL connection. However, in the case where the Secure Socket Layer (SSL) connection is not established, and the XML encryption within the SOAP messages may not cover the whole communication protection. This therefore calls for a means of securing the communications between the users and the web-service servers, thereby ensuring confidentiality of the entire SOAP message.

A user may under different circumstances be required by the web-service server to supply their identification data. This may occur in two situations:

First, an authorised user or a customer of a paid-for weather forecast web-service needs to prove his identity before he or she can be offered the weather service. This customer or user information is transmitted over an SSL connection presumably, so there is guarantee of the communication security, however, there is need for the customer to prove that the web-service server is indeed the correct intended recipient. This may not be possible currently because there is no provision for mutual authentication in the current infrastructure.

Second, a web-service user may under a given situation need to access different web-service applications at the same time. For example, for Nikita, she might need to view her billing information, make an online reservation for her travel ticket, and access the weather forecast web-service at one time logon. In case Nikita is registered for the online reservation, this may require her to authenticate herself each time she needs to use the service. Since the Nikita's billing information is her personal data, not everyone should be able to gain access to it. In that case, she needs to be authenticated before access is granted to her to view the billing information. With the current infrastructure, for every web-service that Nikita accesses, there is no possibility to have one time authentication, which means use of multiple authentications. Therefore, multiple authentications do not provide for reliability and monitoring, it is important that SSO is used within the current SSO infrastructure. As represented in Figure 3.1, a summary of the key components and interactions of Nikita with the web-service servers is given.

The above mentioned situation problems led to the development of security requirements for which Kerberos authentication service is a possible solution to securing web-services with enhanced SSO.

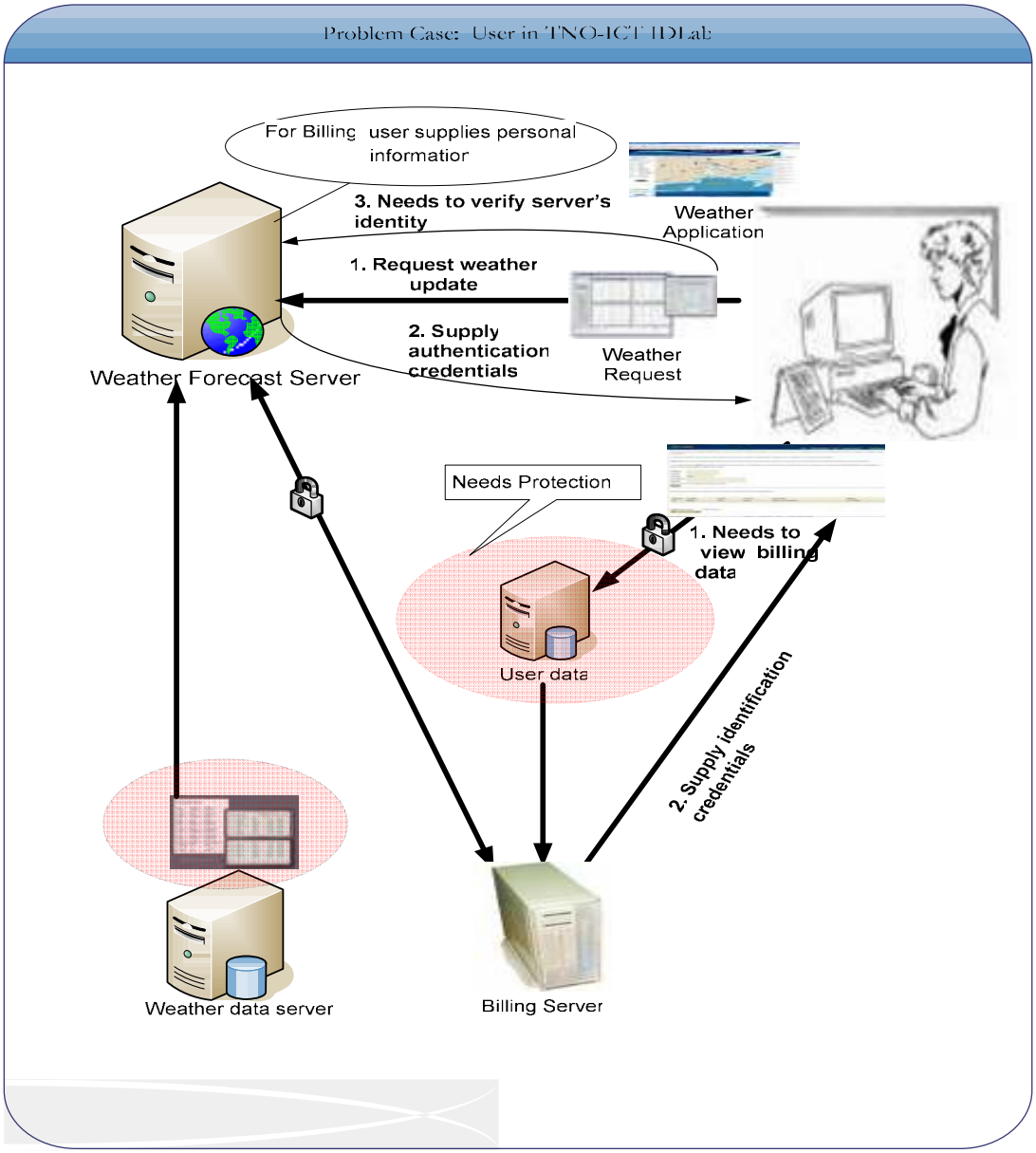

Figure 3.1 Problem description

3.2 Typical SOAP messages in web service calls

In this scenario, Nikita may initiate SOAP messages to the web-service server. The SOAP messages may be protected using the XML Encryption for confidentiality. However, not all

parts of the SOAP message are protected. In that case, when the SOAP message is intercepted, confidentiality of the message in the communication may be compromised. A typical SOAP message that is susceptible to compromise over unprotected communication may be in the form as that shown below. Furthermore, for Nikita to access web-services, she needs a web-service application that contains a SOAP engine, XML converter like SAX (Simple API for XML).

Figure 3.1 shows communication between Nikita and the weather forecast server, through the weather forecast application; and communication to the billing server through billing information request application. Nikita's billing information may include: services paid for by Nikita, account balance, and physical address. This data is sensitive which requires to be protected since it is stored centrally on the billing server. The figure indicates that for all the web-services requested by Nikita, there is need for authentication. Hence, the multiple authentication may not be well managed and monitored which provokes the design of a SSO service for managing web-services.

A typical interaction of Nikita with the web-service servers follows the following sequence:
First, Nikita creates and sends a SOAP request, in XML format over HTTP to the weather forecast server. The generated SOAP Message Request may be in the form of an XML based SOAP message as shown in Table 3.1.

Second, upon submission of user request, one of the server functions is invoked to immediately consult the weather data database to supply the requested information.
Next, the request is received by the web-service server at the weather forecast server; the data is made available to the user in XML format. The response is sent back to the

requesting user. A typical SOAP response message would be in the form of an XML based SOAP message as shown in Table 3.2.

```
POST /WeatherForecast HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body xmlns:m="http://www.example.org/WeatherForecast">
 <m:GetWeatherTemperature>
        <m:PointLatitute>40.00</m:PointLatitude>
        <m:PointLongitude>-80.00</m:PointLongitude>
<temperature type='maximum' units="Fahrenheit" time-layout="k-p24h-7-  1">
      <name>Daily Maximum Temperature</name>
        </m:GetWeatherTemperature>
  </soap:Body>
 </soap:Envelope>
```

Table 3.1 SOAP Message Request

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/WeatherForecast">
    <m:GetWeatherTemperatureResponse>
      <m:Temperature>79</m:Temperature>
    </m:GetWeatherTemperatureResponse>
  </soap:Body>
</soap:Envelope>
```

Table 3.2 SOAP message Response

Finally, to the user, communication is successful when the user receives a response from the web-service provider. However, for a security conscious user, a communication must be both successful and authentic through mutual authentication.

Based on the communication of our case user, security requirements have been identified as listed in the next section.

3.3 Security requirements

There are two stakeholders to be considered, thus the users of the web-services and all other services and the service providers (either web-services or any other service). We describe the user requirements and the web-service server requirements envisaged. These requirements have been used for developing the architecture described in chapter four.

(a) User Security requirements:

Based on the analysis of the web-service scenario with our typical user Nikita, the following user security needs were identified:

1. The user requires his or her data to be protected thus ensure confidentiality,
2. The user needs the servers providing the services to be authenticated, hence be known to the user,
3. The user needs to have SSO for user access to numerous web-services with a single login or authentication data.
4. The user requires correct weather forecast data to be returned and this applies to data from other service providers.

(b) Web-service server security requirements:

Based on the analysis on the user web-service interactions, the following web-service server security needs were identified:

1. The user must show proof of identity to the service providers, through authentication
2. The service provider's sensitive data must be kept confidential.
3. The weather service server that offers paid-for weather forecast services needs to guarantee that the requesting user has paid-for the service. Also, where a bill is to be

directed to the user, then it is mandatory for the user to identify him or herself to the server.

3.4 Review

The analysis was used to develop security requirements that were considered in the design of the architecture for securing web-services through Kerberos service. In chapter four, we describe how the concept of Kerberos was adopted for securing web-service communication and delivering SSO.

Chapter 4: System Design

4.1 Introduction

In this chapter, how Kerberos authentication service was used to address the SSO need in the web-service environment, while delivering communication security. The services that Kerberos offers to ensure web-service security are described. Based on the system requirements described in chapter three, the architecture was designed based on Kerberos authentication service.

Furthermore, the architectural design in Figure 4.1 was inspired by the fact that currently web-service communications are not centrally managed in terms of authentication. To enable central authentication of web-services with SSO, we introduced one service principal (the domain-relay) which understands the Kerberos protocol. As an addition, an application (the application-client) was also introduced, which resides at the client machine and the server-end that was used to intercept the SOAP messages. The domain-relay service principal is located at both the client side and the server side, in that it can take on roles of client or server depending on the initial origin of the SOAP messages. A description of the roles the service principal and the application-client play are described in detail in the current chapter. A number of assumptions were made to govern the architectural design infrastructure for its proper functioning and service delivery.

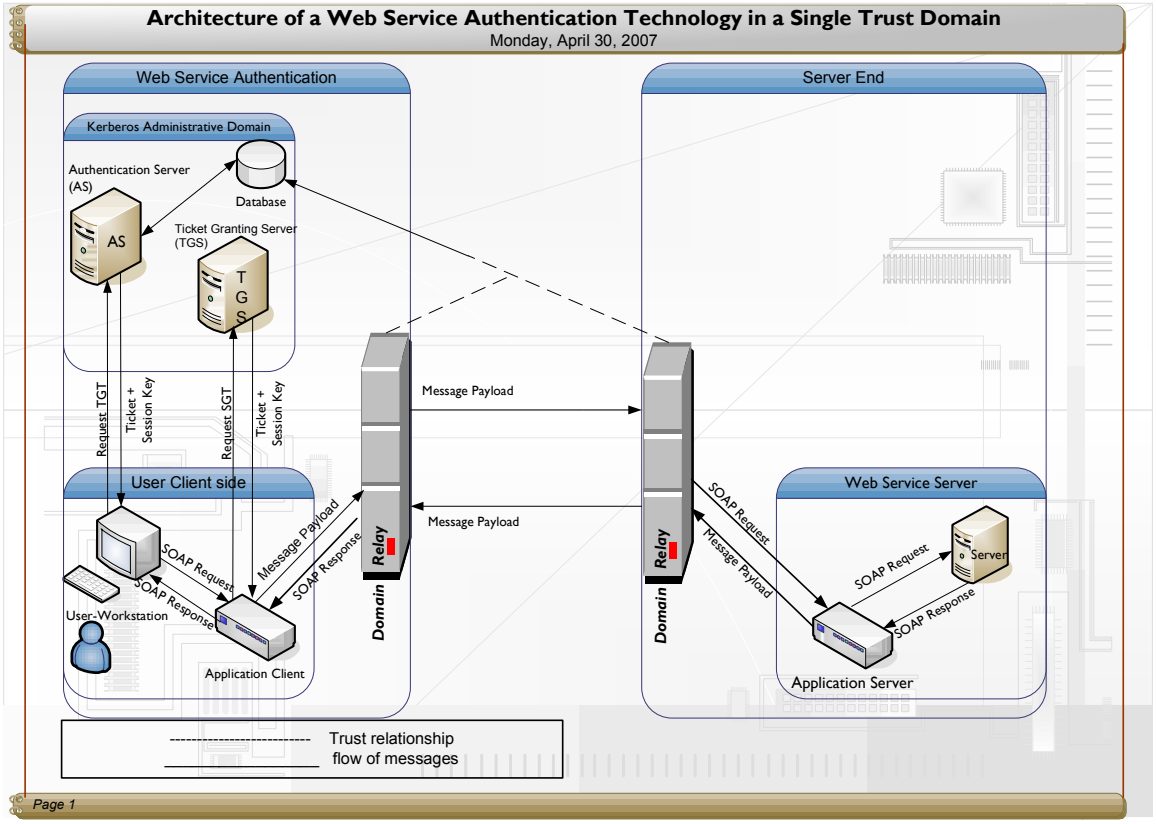

Figure 4.1 Architecture of authentication in a web service environment

Assumptions

A number of assumptions were made that include the following:

1. The web-service environment is constrained to a single trust domain and all the web-services are within the Kerberos environment.
2. The administrator is given a single password that is secure and stored safely. The administrator uses his username and password for authentication and registration of web-services to the Kerberos domain.
3. The Kerberos server maintains a database of users, domain-relay service principal (both at client side and server side), application-client (both at client side and server side), and the web-service servers.

4. The user's client or workstation initiates the request for a ticket granting ticket as in typical Kerberos. However, since the communication of web-services is done through SOAP, then we also assume that the application-client intercepts the SOAP requests, and submits to the domain-relay.
5. For reliable accessibility of web-services, both the Kerberos server and the domain-relay must have backups.

Overall, the designed architecture makes use of the domain-relay (developed concept) as a service principal, since it is aware of Kerberos. In addition, other participants in the architecture include among others the application-client (developed concept), users and web-services. Through the domain-relay, the web-service servers do not need to be knowledgeable with Kerberos. Besides, in order to allow for numerous client service applications to use a single connection to the domain-relay, the application-client is used. The service principal and the other participating entities are described in detail in the next section, followed by the description of the communication flow.

4.2 Description of the service principal and the participating entities

Based on the architecture described in Figure 4.1, the services offered by the service principal and other participating entities is described in this section. The description of the user, domain-relay, application-client and the web-service servers is given. The user is a principal as in the Kerberos domain, since in the original Kerberos protocol, a user within the Kerberos domain is created as a principal (someone who requests for the services in the Kerberos domain). In our context, the application-client (software program running in the client space) takes the role of user and is responsible for intercepting the web-service requests and sending them on behalf of the user. In principle, the application-client has

access to all the files and applications in the client machine that are necessary for carrying out a web-service transaction. Therefore, all applications connect to the application in case any transaction is to be carried out. The domain-relay is responsible for vouching the identity of the application-client and also guarantees the confidentiality of the web-service communications. As a result of the use of Kerberos service, a client-server relationship is established between the domain-relay and the application-client. Lastly, the *URL* of the web-service server is used for registration in the Kerberos database as an identifier. However, the web-service servers are not knowledgeable of the Kerberos protocol because it avoids the need to extend each web-service with a code to learn the Kerberos protocol.

If web-services are to learn the Kerberos protocol, it would require that a piece of code is introduced for each web-service running at both the user client-end and the server-end. This piece of software code would be used to extend the web-services to learn the Kerberos protocol and make use of it for their communications. The problematic and daunting task on the software implementer's part is that different pieces of code are needed for all the different services available. Therefore, to avoid going through this kind of hassle, we introduce the application-client is introduced. The application-client was designed to be interoperable with all web-services such that all web-services easily make use of it. The application-client accesses the files and client data including the tickets and user client credentials to be used for authentication. In so doing, several web-services that are started by the user are all intercepted by a single application-client running in the user client machine. The application-client, being a software program, running in the user client machine, assumes the identity of the user principal whenever it relays the SOAP requests to the domain-relay. The Kerberos service infrastructure plays a role between the user client-end

and the server-end. The services offered by the service principal and the participating entities through the Kerberos service architecture are now described. In the new architecture, N – client users need exactly N - application-clients since all the web-services need a single application-client. The N - application-clients, all make use of a single domain-relay within a single trust domain.

4.3 Services

In the new architecture, we assume that the user principal is an existing principal in a single trust domain that uses Kerberos authentication service. In that case, the user has identification credentials used for authentication which are known to the Kerberos authentication server. When the user needs a specific service in the Kerberos domain, he or she needs to request for the TGT that is used for getting the service ticket which is later used to gain access to a specific service. However, if a web-service is requested, then the application-client assumes the identity of the user. In this case, when the web-service call is made, it is intercepted by the application-client and relayed to the domain-relay such that it appears to come from the user directly.

Therefore, the services the application-client and the domain-relay in the new architecture offer ensure security in the web-service communications as well as SSO. The role of being a client or server, for the case of the application-client and domain-relay, is dependent on whether a synchronous or an asynchronous web-service call took place. Hence, the services offered by the application-client and the domain-relay are described in terms of whether a synchronous or an asynchronous communication was carried out at both the user client-end and the server-end. In the context of web-service calls, a synchronous communication is one

where the web-service request is followed by the response immediately. On the other hand, an asynchronous communication is one where there may be a delay in responding to the web-service request which means as soon as the response is ready. The web-service servers sending the messages take on the role of client at that point in time. The description given in the next section is about services and roles played by the web-service application, application-client, domain-relay and web-service servers at the user client-end and later at the server-end.

4.3.1 User - Client end

The web service application**:**

The web-service application installed at the user client-end enables the user to make use of the web-services that he has subscribed for. The web-service application is able to function properly when a SOAP engine and the XML converter like SAX (Simple API for XML) are installed. Through the web-service application, SOAP message requests are formulated intended for specific web-service servers through the URL indicated. Since the web-service application is not knowledgeable of Kerberos and the communication and the messages need to be protected, the messages from the web-service application are intercepted by the application-client. When the messages are intercepted, the confidentiality of sensitive data is guaranteed, since from this point on the application-client uses the Kerberos protocol to ensure protection of the messages and the communication.

In the current new architecture, the SOAP messages originating from the web-service application are set to pass through the application-client. Between the web-service application and the application-client, messages are protected by the operating system. In

this case, there is authentication between the web-service application and the application-client. In this way, the messages are protected by the connection established, through SSL service. As soon as the messages are received by the application-client the rest of the message protection is done through Kerberos. This is described in the sub-section below.

Application-client:

In this new architecture, web-service calls are initiated by the user and later intercepted by the application-client since web-service application does not need to learn the Kerberos protocol. Recall from chapter three, one of the user's security requirements, was that the user needed to interact with an authentic web-service server. They need to have proof that responses are coming from the correct and intended web-service server. In this case, the server should prove their identity to the user. So, the application-client ensures that the messages are directed to the domain-relay since there is a trust established between the application-client and the domain-relay.

The application-client, being a software component running in the user client-end, is responsible for ensuring that all messages that originate from the different web-service applications are intercepted, and protected before being sent to the domain-relay. Through the web-service applications, the user is able to initiate the web-service requests. Therefore, messages from the web-service applications are directed to the application-client. Communication between the web-service application and the application-client is protected through the operating system security because at this point the messages may be susceptible to loss of confidentiality and integrity. The authentication within the operating system

supports that messages are directed only between the authentic web-service applications and application-clients.

In new architecture, the application-client is introduced as an intermediary software process that can learn the Kerberos protocol. In addition, the application-client is used to address the problem of creating separate software applications required for incorporating Kerberos protocol in the different web-services. Therefore, for (N) users, there will be (N) application-clients. All these application-clients make use of a single domain-relay within their single trust domain.

To ensure that the domain-relay authenticates the application-client, the application-client takes on the identity of the user and it uses this for further communication with the domain-relay. The application-client assumes the identity of the user and it uses this identity for authenticating itself to the domain-relay. Between the application-client and the domain-relay, data protection and authentication is guaranteed. This is because there is a Kerberos client-server relationship between the application-client and the domain-relay.

Kerberos security protocol offers mutual authentication whereby the application-client authenticates itself to the domain-relay and vice versa. This is only possible because within Kerberos, a client-server relationship ensures that a message originating from the application-client is only directed to the intended domain-relay and not any other software program masquerading as the domain-relay. Since both the application-client and the domain-relay share each other's identities, this ensures that the messages from the application-client are only directed to the known domain-relay. Therefore, for any other

domain-relay which is not known to the application-client (masquerading) will not receive the messages from application-client. This kind of security restriction is only possible through Kerberos security. The messages originating from the application-client are in form a Kerberos message payload. The messages are in encrypted form and hence not understandable, and the whole communication is also protected through Kerberos. The message payload basically contains the SOAP request and the URL of the intended web-service server and the ticket containing the identity of the application-client.

Furthermore, due to the need for the user to start up different web-service requests concurrently, through Kerberos authentication service, it is possible for the application-client to receive (M) different message requests, as the port that intercepts the port is configured to listen to numerous messages. The application-client only needs the URL of the intended web-service recipient to be used by the domain-relay to determine the destination of the message.

Practically, the application-client simply requests the domain-relay to deliver the messages that it passes to it, to the URL of intended web-service server, but most importantly is that the application-client makes sure that web-service call is not directed anywhere else but to the correct domain-relay. On the other hand, the domain-relay knows who is asking for the service. This is possible because of the Kerberos client-server relationship established. The domain-relay is a service principal that provides services to the application-client. The services provided by the domain-relay are described in the subsequent section.

The domain-relay:

The domain-relay at server-end ensures delivery of the SOAP messages to the intended recipients, through an established Kerberos client-server relationship between the domain-relays, because these domain-relays at client and server-ends are service principals within the Kerberos database. The domain-relays play client or server roles depending on whether it is synchronous or asynchronous communication.

For synchronous communication the domain-relay plays the role of server for the application-client and domain-relay relationship. In this communication, based on our user case, the security consideration would be for the user to be authenticated to the domain-relay server. This is because the domain-relay at the server-end needs to know who the requesting user is. Therefore, the domain-relay basically vouches the identity of the application-client to the domain-relay at the server-end. Hence the domain-relay transmits the web-service calls. The message payload at this point will contain the message payload from the application-client and userID of the application-client.

For asynchronous communication the domain-relay plays the role of server. The server security considerations show that the domain-relay must deliver the SOAP messages to authentic user. In this case the domain-relay only needs to know the userID of the intended user recipient. The userID is obtained from the message payload originating from the domain-relay at the server side. Furthermore, from the user's point of view, it is important that the servers authenticate themselves to the user. This is achieved when the domain-relay at the server side vouches the identity of its application-client which bares the URL as its identification. The trust between the two domain-relays is assured because of the existing

Kerberos client-server relationship. This relationship also ensures that there is an establishment of mutual authentication. Also at this point, there is confidentiality Kerberos security protocol which is used for encryption and protection of the whole communication. The message payload at this point contains the previous message payload from the domain-relay at the server side and the application-client.

4.3.2 Server - end

The domain-relay:

For synchronous communication, the domain-relay plays the role of server. In reference to the security considerations from the user case, it is important for the domain-relay that an authentic user requests for the services. Through Kerberos client-server relationship between the domain-relay and the application-client, the above security considerations can be achieved. First, the message payload from the synchronous communication is received from the domain-relay at the user client side. The userID of the application-client who requests is presented as a form of authentication. Furthermore, to ensure that the web-service SOAP messages are received by the intended web-service server, the domain-relay directs the SOAP request to the application-client. The application-client is trusted by the domain-relay. In addition, the application-client knows the identities of the different URLs because the application-client is running on the server side.

In that case, the application-client is knowledgeable of all the URLs that have been added to the Kerberos domain. Hence, a message payload is directed from the domain-relay at the user client side to the domain-relay at the server side. The message payload contains the message payload from the application-client and the userID of the application-client. In

asynchronous communication, as per the user case security considerations, there is need for the servers to be authenticated to the user and that confidentiality of the SOAP responses must be guaranteed.

Application-client:

Usually the application-client is aware of the identities of the URLs; which is used for mutual authentication to the user when the web-service response is finally received. The domain-relay vouches the identity of the application-client to the domain-relay at the user client-end. Since there is a Kerberos client-server relationship between the domain-relay at the server side and the domain-relay at the user client side, this relationship ensures that mutual authentication is established. In this case the messages can be guaranteed to be finally delivered to the intended users. Also, there is assurance of confidentiality as a service offered by the Kerberos protocol. The application-client assumes the identity of URL while responding to the web-service calls since it has the capability for SSO. These requests are assumed to be in a synchronous communication or upon initiation of an asynchronous communication. The role of this application-client is that it should be Kerberos aware such that the web-service servers do not need to learn the Kerberos protocol.

During the asynchronous communication, SSO is achieved through Kerberos authentication service, which allows all the web-service servers to start responses to the previously requested services without the application-client being dedicated to just a single web-service server. Therefore all the web-service servers in the domain have equal access to the application-client without causing conflict. Furthermore, protection between the application-client and web-service servers is guaranteed by protection from the Kerberos security service.

Between the web-service server and the application-client, a SOAP message response is in transit. The message response is sent as a Kerberos message payload between the application-client and the domain-relay. Precisely, the message payload would contain the SOAP message response and the userID of the intended recipient. In doing so, the application-client can achieve single sign on as well as guarantee that the web-service servers do not have to learn the Kerberos security protocol. Within the server side, authorization to the different services is done using the criteria described in the next section.

4.4 Communication flow in the new architecture

The new architecture is a communication process from the time of authentication through to the time of authorization. The protocol is shown in Figure 4.2., which is a representation of the flow of data between the service principal and the participating entities. A set of identifiers that are used in the communication is described together with the ticket exchange.

4.4.1 Authentication

The authentication process starts after the user has already acquired the TGT. The user is only left with getting the service ticket. The user client at this point initiates a web-service call, which may be a SOAP message meant for a specific web-service. The purpose of the SSL connection between the application-client and the domain-relay is to ensure the security of the message payload in transit. The SOAP message is intercepted by the application-client and redirected to the domain-relay. The application-client takes the SOAP message and appends the authenticator; this message is turned into a payload to be sent to the domain-relay. The domain-relay trusts the application-client to provide the correct and

authentic identification data. The domain-relay initiating the communication is usually the client while the receiving one is the server.

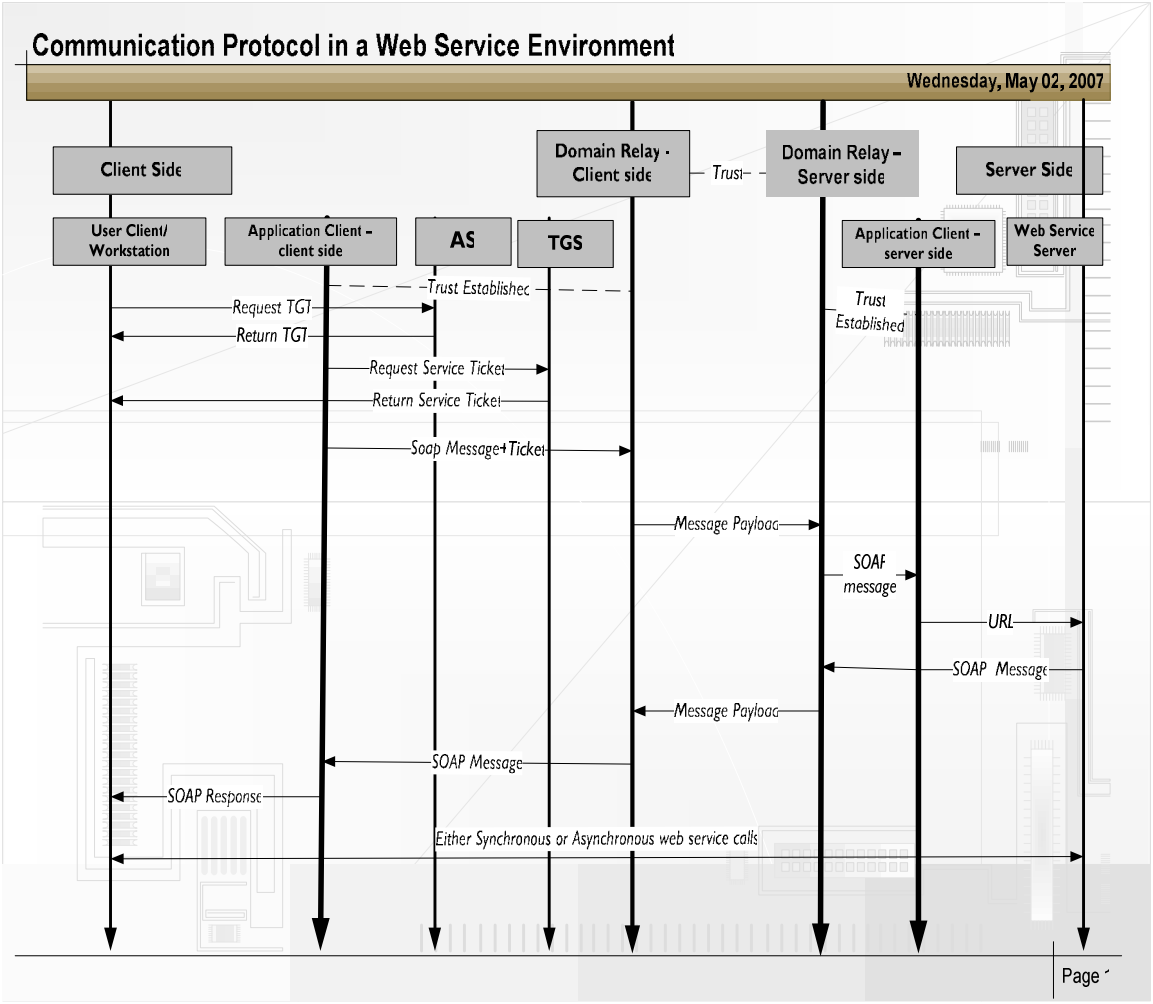

Figure 4.2 Communication Protocol

As soon as the message payload arrives at the domain-relay at the client side, the SOAP message is recovered and the authenticator is opened for identity verification. The userID is used by the domain-relay to vouch for the identity of the application-client to the domain-relay at the server side. If several message payloads are received, then the domain-relay at the client-end vouches for different identities to the domain-relay at the server side. In so doing, SSO is made possible. As soon as the domain-relay at the server-end receives the message

payload, it opens it and retrieves the URL of the intended web-service. The domain-relay makes sure that only authentic application-client at the server-end receives the specified URL.

This is made possible because there is a Kerberos client-server relationship between the domain-relay and the application-client. Therefore, the application-client at the server-end checks through the registered web-services to confirm the existence of the requested web-service. This is made possible through the UDDI that publishes the web-services. The application-client has a port that listens for all the registered and published web-services, and then updates its archive at specified time intervals. When the web-service is discovered, the SOAP request is redirected to the specified web-service server.

The response to the SOAP request depends on whether the communication is synchronous or asynchronous. In this case, the application-client and the domain-relay play different roles. Once the mode of communication is established depending on the readiness of the response, the response process starts from the web-service server. In this case, the web-service server will play a role of client or server depending on whether it is asynchronous or synchronous communication respectively. If mutual authentication is required, then the web-service server also presents its authenticator together with the SOAP response to be used later by the user client to authenticate the source of the response, through the identity presented in the authenticator.

The communication process is also described with indication of the message contents transported during the process. The tickets and message exchanges are also described.

4.4.2 Authentication attributes

The typical Kerberos, Authentication Dialogue includes several attributes that are listed below as described in [STAL03]. These attributes have been adapted for use in the new architecture, together with the additional attributes applicable to the new architecture.

1. C = Client
2. ID_c = Identifier of user on c,
3. AS = Authentication server,
4. TGS = Ticket Granting Server,
5. TGT = Ticket Granting Ticket,
6. P_c = password of user on c,
7. AD_c = Network address of c,
8. K =Secret key shared between principals
9. $Authenticator_c$ = Generated by the client to validate the ticket.
10. $Authenticator_s$ = Generated by the Server for mutual authentication.
11. TS = Timestamp
12. Lifetime
13. URL = Identifier of web-service server
14. AC_c = Application-client – client-end – uses the identity of the client, C
15. AC_s = Application-client – server-end - uses the identity of the server, *URL*
16. DR_c = Domain-relay – client-end
17. DR_s = Domain-relay – server-end,
18. ‖ = concatenation

Table 4.1 Adapted authentication dialogue attributes in the new architecture

4.4.3 Session Key exchange

In typical Kerberos, a secret is maintained between two communicating principals. When the client, C requests for a TGT, a session key is shared between the user client and the TGS. The authentication server generates an encryption key (using the secret key of the client, thus a one-way hash of the username and password) which is used for encrypting the TGT and the session key sent to the user client, C. Furthermore, another session key is generated by the TGS and shared between the user client and the domain-relay. This session key is encrypted using the user client and TGS session from the previous session key. These keys are shared in local cache in the client or workstation.

However, for the new architecture, an additional session key is generated and shared between the domain-relay at the user client-end and the domain-relay at the server-end to ensure protection of their communications. Like in normal Kerberos, the user client requests for a TGT and the encryption key is represented as $K_{c,tgs}$. This key is used by the application-client when sending the SOAP requests to the domain-relay, with the identity of the user client. The authenticator is encrypted within this key to be used by the domain-relay at the client-end to prove that the message is authentic.

Hence, the tickets are mainly used for carrying the session keys which are used during authentication process and between the application-client and the domain-relay at both the client side and the server side. The keys are stored in the client cache and accessed by only the eligible user upon request as the need arises.

4.4.4 Authentication Message Exchange

The TGT issued by the AS is required to get access to the TGS. Once that access is granted, the TGS issues a service ticket that allows access to various web-services using the following defined attributes used during ticket exchange.

$Ticket_{tgs}$ = *A ticket offered to the client by the AS allows for access to a TGS.*
$Ticket_{s}$ = *Ticket used by the client to gain access to the web service server through the domain-relay at server end, and is offered by the TGS.*
Ek = Encryption key (may be a secret key between the user client, domain-relay at client end, and the TGS or may be a session key of user client, TGS and domain-relay at user client end).
TS = Indicates time the ticket was issue d.
Lifetime = Prevents replay attack by showing the time allowed for the ticket to be used.
$K_{c,tgs}$ = *Session key shared by the client and TGS, offered by the AS.*
$K_{c,s}$ = *Session key shared by client and server, offered by the TGS.*

Table 4.2 Definition of Attributes used in Message Exchange

The user client applies for the TGT from the AS upon logon to the client. This phase is always transparent to the user at the client or workstation. The AS responds with the ticket as shown in Messages A, B and the TGT is used to request for a service ticket from the TGS. The TGS responds as shown in messages C and D as indicated in the Table 4.2.

AS Message Exchange

A. C → AS: $ID_c \| ID_{tgs} \| DR_c \| TS_1$

B. AS→C: $Ek_c [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}]$

$Ticket_{tgs} = Ek_{tgs} [K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2]$

Service Ticket Message Exchange

C. C → TGS: $V \| Ticket_{tgs} \| Authenticator_c$

$Authenticator_c$ **=** $EK_{c,tgs} [ID_c \| AD_c \| TS_3]$

D. TGS → C: $Ek_{c,tgs} [K_{c,v} \| ID_v \| TS_4 \| Ticket_v]$

$Ticket_s = Ek_c [K_{c,tgs} ID_c \| AD_c \| ID_{tgs} \| DR_s \| DR_c \| TS_4 \| Lifetime_4]$

Application client Message Exchange

E. C→ AC_c: SOAP Message

F. AC_c→ DR_c: SOAP Message $\| Authenticator_c$

Domain-relay Message Exchange

G. DR_c→ DR_s: Message Payload

H. DR_s → AC_s: SOAP Message

I. AC_s → URL: SOAP Request

Message Response from Web Service Server

J. URL → AC_s: SOAP Response

K. AC_s → DR_s: [SOAP Message $\| Authenticator_s$]: ***$Authenticator_s$*** **=** $EK_{c,v} [TS_3 + 1]$

L. DR_s → DR_c: Message Payload

M. DR_c → AC_c: SOAP Message

N. AC_c → C: SOAP Response

Table 4.3 Message Exchange in the architecture

The application-client, denoted by AC_c takes the intercepted SOAP message together with the $Authenticator_c$ which is a message payload and forwards it to the domain-relay, denoted by DR_c. The DR_c checks the $Authenticator_c$ as prove that the client requesting the service is authentic. This is done by comparing ID_c in the $Authenticator_c$ and that in the ticket, $Ticket_s$. The DR_c vouches the user identity to the DR_s at the server-end. The DR_s redirects the validated SOAP message to the AC_s whose duty is to identify the web-service server through the URL provided. The web-service server performs the authorization of the SOAP requests, on whether to respond or not. Therefore, the authorization of the SOAP requests is beyond the offering of the new architecture, once the SOAP request is delivered to the specified web-service server. [HILL05] stated that currently, web-services in organizations are protected by web-service firewalls. The firewalls address the issue of access control (authorization), thus distinguishing between proper access and improper access but not allowed or disallowed clients which resolves the authentication issue. Hence our major concern was on authentication in the web-service environment through Kerberos, and enhanced SSO.

When the response is sent back to the client either immediately or at a later time, the client is only able to see the contents of the message by decrypting the message contents, through the shared key between the client and the server. It is only the right client who can decrypt the message. In this case, the whole communication can not be tampered with since the messages are protected through Kerberos security.

4.5 Review

A design of the new architecture for securing web services is described. The new architecture achieves SSO and communication security within the web service environment. This architecture can be adapted for use in a setting where web services are used. The new architecture accomplishes its functioning through use of Kerberos authentication service, the application client and the domain relay. The next chapter gives the discussion, conclusion and recommendations for the study.

Chapter 5: Discussion, Conclusions and Recommendations

5.1 Discussion

The primary purpose of this research was to investigate the feasibility of applying the Kerberos security for use within a web-service environment in order to enhance SSO as well as security. A secondary purpose was to investigate the current SSO mechanisms used within a single trust domain, and their applicability for web-service security. To be specific, the research was designed to address the following research questions.

1. How is SSO authentication handled in web-services?
2. What is the current authentication challenge faced within a web-service environment? And how can they be resolved?
3. How can the Kerberos concept be applied within a web-service environment to achieve SSO authentication?
4. Is it possible to adapt the Kerberos concept for SSO in a multiple trust domain of web-services

For each of these questions, a summary of the findings and discussion is now presented.

Due to the growing use of web-services, there was emergence of inter and intra domain communications. This meant re-authentications of the users each time a service was required. Therefore as per the project by Liberty Alliance [ROSE04], a mechanism was established that made use of SAML authentication assertions across multiple security domains. In this case, the identification credentials are shared across domains. Therefore,

this level of shared identification credentials did not necessarily lead to access to various applications without re-authentication. This indicated that SSO was still lacking although mechanisms to share identification credentials were already established.

Furthermore, web-service users faced a number of challenges such as loss of confidentiality in the communication since XML security is limited to message security as in [ROSE04, GRAH04]. In addition, there is a method that was established for mutual authentication, and yet this was a crucial element when dealing with sensitive data. Besides, users accessing various web-service applications needed to be authenticated each time they requested a service. Hence the lack of SSO capability in web-services creates a challenge on how fast and manageable web-services are. These challenges were addressed by adoption of Kerberos security concept in the web-service environment.

Kerberos authentication service has been used in windows domain authentication and it has proven to protect communications as well as offer SSO. The Kerberos concept could only be used within the web-service environment through additional applications that have been designed as new concepts. These applications were the application-client and the domain-relay which were introduced to act as intermediaries in the web-service environment. This is because whereas Kerberos protocol is based on TCP/IP protocol, while the web-services are based on the SOAP protocol. A description of both the application-client and the domain-relay, shows that the application-client understands both the SOAP protocol as well as the TCP/IP protocol. On one hand it helps in sending the SOAP messages over the TCP/IP communication. On the other hand, the domain-relay is the heart of the new architecture since it is fully knowledgeable of Kerberos and has a trust with the application-client. The application-client trusts the Domain-relay to provide the secure services.

Therefore, the new architecture designed addressed the SSO problem and communication security through Kerberos. This is achieved through the application-client and the domain-relay. The application-client is a software program that is set to intercept the SOAP requests originating from the client. The application-client protects these SOAP messages through Kerberos tickets and forwards this message to the domain-relay at the client-end. The domain-relay vouches the identity of the application-client to the domain-relay at the server-end. In so doing, a trust is established between the domain-relay - client-end and domain-relay – server-end assists in ensuring that the communication is protected. The SSO capability is achieved because the application-client has the ability to receive so many web-service requests at the same time and all these can be protected and viewed without need for re-authentication.

5.2 Conclusions

This study has provided insights into the challenges faced by web-service user within the web-service environment. It has also shown the different technologies that are beings used for web-service security. Some of these technologies include among others Kerberos authentication service. The Kerberos authentication service was adapted for use within the web-service environment for achieving SSO and securing communications. Therefore, the designed new architecture seems a partial solution that can be used to address the SSO and communication challenge.

Furthermore, it was found that web-services within a single trust domain could be protected easily in the first phase of the design. However, for the multiple trust domains, the new architecture could be extended. This is a possible area for further study, since multiple trust domains are very complex and dynamic in their nature. Therefore, because the initial focus

was on a single trust domain, the result of the new architecture has indicated that the Kerberos concept could be used.

5.3 Limitations

There are a couple of limitations in this research that may have influenced the current result. Firstly, the time available was not enough to address all the research questions. We had hoped to make a study of extending the current design to a multiple trust domain. This would have been possible, only if we had the new architecture implemented as a proof of the concept. However, the implementation was not done due to time limitation. Therefore, the current architectural design is only limited to a single trust domain.

Secondly, the area of study needed a lot of expert programming knowledge in the web-service area. This was lacking initially since at the beginning of the study, the area was a new concept to the principal researcher and so required lots of reading to be done to understand the topic and its intricacies.

5.4 Recommendations

Based on the findings and discussion of the results of this study, the following recommendations are considered in two parts (1) for the companies and organisations, and (2) for further research.

5.4.1 For Companies and Organisations

1. Because the designed new architecture was developed based upon use case scenario of a company, TNO-ICT. This new architecture should be used in any organisational or company platform, since the domains are more or less the same.
2. Given that no research answers all questions, it means the new architecture at this moment may be limited to the challenges within the TNO-ICT situation.
3. For any company or organisation wishing to protect their web-services, this new architecture is recommended to be adapted in their context because the concept is the same for all companies and organisations.
4. Companies and organisations should now be able to protect their web-service communications as well as achieve SSO using this new architecture.

5.4.2 For Further Research

For the benefit of other researchers and interested parties in this area, further research and extension studies are recommended in:

1. A study on how the Kerberos concept can be extended for use within a multiple trust domain and how security can be achieved in this kind of domain.
2. An investigation of ways in which SSO can be achieved in the web-service environment with other security concepts used.

In conclusion, because web-services are currently in wide use and their security is paramount, this study has contributed to our knowledge of how SSO is achieved in a web service environment and researchers and organizations should continuously, think of novel ideas on achieving total security in their workplace.

References

BEAW05 BEAWebLogic Enterprise Security™®. *Administration Application Guide*. Copyright © 2005 BEA Systems, Inc. Version 4.2 Service Pack 2.

CLER02 Clerq, J. D. 'Single Sign-On Architectures' in: *Proceedings of the International Conference on Infrastructure Security, 2002,* pp. 40 – 58, ISBN: 3-540-44309-6.

DELP02 A Delphi Group White Paper. Web-services. Market Milestone Report. Santa Clara: Sun Microsystems, Inc., 2002.

ERL04 Erl, T. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Upper Saddle River, NJ: Prentice Hall, 2004.

GOTZ04 Gotze, J. *Single Sign On in Web Service Scenarios*. AG Integrierte Kommunikationssysteme. Kaiserslautern: TU Kaiserslautern, 2004.

GRAH04 Graham, S., Davis, D., Simeonov, S., Daniels, G., Brittenham, P., Nakamura,Y., Fremantle, P., König, D.,and Zentner, C. *Building Web Services with Java. Making sense of XML, SOAP, WSDL, AND UDDI*, 2nd Ed. Indianapolis, Indiana: Sams Publishing, 2004.

HILL05 Hillenbrand, M., Götze, J., Müller, J. and Müller, P. 'A Single Sign-On Framework for Web-Services-based Distributed Applications' in: *8th International Conference on Telecommunications ConTEL, Zagreb, Croatia,* June 2005.

IBM03 IBM and Microsoft Corporation. Federation of Identities in a Web-services World. Version 1.0, 2003. At: ftp://www6.software.ibm.com/software/developer/library/ws-fedworld.pdf [20/03/2006].

IBM03a IBM Corporation, Microsoft Corporation, BEA Systems, Inc.., RSA Security, Inc.., Verisign, Inc.. Web-services Federation Language (WS-Federation), 2003. At: ftp://www6.software.ibm.com/software/developer/library/ws-fed.pdf [20/03/2006].

IBM03b IBM Corporation, Microsoft Corporation. Web-services Security Kerberos Binding, 2003. At: http://www.cgisecurity.com/ws/WSS-Kerberos.pdf [16/03/2006].

KHAN03 Khan, F. Design secure client/server Java applications that use GSS-API and Kerberos tickets to implement SSO, 2003. At: http://www-128.ibm.com/developerworks/java/library/j-gss-sso/ [25/04/2006].

MAcD05 MacDonald, J. and Mitchell, C. J. 'Using the GSM/UMTS SIM to Secure Web-services' in: *Proceedings of the 2nd Workshop on Mobile Commerce and Services (WMCS 2005), Munich, July 2005*.

NEWC05 Newcomer, E. and Lomow, G. *Understanding SOA with Web Services*. Upper Saddle River, NJ: Prentice Hall, 2005.

O'NEI03 O'Neill, M. *Web Services Security*. Berkeley, California: The McGraw-Hill Co, 2003.

PASH03 Pashalidis, A. and Mitchell, C. 'Using GSM/UMTS for Single Sign-On' in: *Proceedings of SympoTIC '03, Joint IST Workshop on Mobile Future and Symposium on Trends in Communications, Bratislava, Slovakia, October 2003*, IEEE Press, 2003, pp. 138-145.

PLEE03 Pfleeger, C. P. and Pfleeger, S. L. *Security in Computing*, 3rd Ed. Upper Saddle River, NJ: Prentice Hall, 2003.

RFC4537 Zhu, L., Leach, P. and Jaganathan, K. Request for Comments 4537. *Kerberos Cryptosystem Negotiation Extension.* Network Working Group. June 2006

RFC4120 Neuman, C., Yu, T. and Raeburn, K. Request for Comments 4120. *The Kerberos Network Authentication Service* (V5). Network Working Group. July 2005.

RFC1510 Kohl, J., Digital Equipment Corporation, and Neuman, C. Request for Comments 1510. *The Kerberos Network Authentication Service* (V5). Network Working Group. September 1993

ROSE04 Rosenberg, J. and Remy, D. *Securing Web Services with WS-Security. Demystifying WS-Security, WS-Policy. SAML, XML Signature, and XML Encryption.* Indianapolis, Indiana: Sams Publishing, 2004.

STAL03 Stallings, W. *Network Security Essentials. Application and Standards*, 2nd Ed. Upper Saddle River, NJ: Prentice Hall, 2003.

STEE05 Steel, C., Nagappan, R. and Lai, R. *Core Security Patterns. Best Practices and Strategies for J2EE™, Web Services, and Identity Management.* Upper Saddle River, NJ: Prentice Hall, 2005.

SURF05 SURFnet bv. A-Select, October 2005. http://aselect.surfnet.nl. Retrieved on the 22nd 2007.

TUNG99 Tung, B. Kerberos. *A Network Authentication System.* Massachusetts: The Addison-Wesley: Longman, Inc., 1999.