Towards Secure Web Services:

Performance Analysis, Decision Making and Steganography Approaches

A thesis submitted for the degree of Doctor of Philosophy

By

Bachar Alrouh

Department of Information Systems and Computing,

Brunel University

October 2011

ABSTRACT

Web services provide a platform neutral and programming language independent technology that supports interoperable machine-to-machine interaction over a network. Clients and other systems interact with Web services using a standardised XML messaging system, such as the Simple Object Access Protocol (SOAP), typically conveyed using HTTP with an XML serialisation in conjunction with other related Web standards. Nevertheless, the idea of applications from different parties communicating together raises a security threat. The challenge of Web services security is to understand and consider the risks of securing a Web-based service depending on the existing security techniques and simultaneously follow evolving standards in order to fill the gap in Web services security. However, the performance of the security mechanisms is fraught with concerns due to additional security contents in SOAP messages, the higher number of message exchanges to establish trust, as well as the extra CPU time to process these additions. As the interaction between service providers and requesters occurs via XML-based SOAP messages, securing Web services tends to make these messages longer than they would be otherwise and consequently requires interpretation by XML parsers on both sides, which reduces the performance of Web services. The work described in this thesis can be broadly divided into three parts, the first of which is studying and comparing the performance of various security profiles applied on a Web service tested with different initial message sizes.

The second part proposes a multi-criteria decision making framework to aid Web services developers and architects in selecting the best suited security profile that satisfies the different requirements of a given application during the development process in a systematic, manageable, and effective way. The proposed framework, based on the Analytical Hierarchy Process (AHP) approach, incorporates not only the security requirements, but also the performance considerations as well as the configuration constraints of these security profiles. The framework is then validated and evaluated using a scenario-driven approach to demonstrate situations where the decision making framework is used to make informed decisions to rank various security profiles in order to select the most suitable one for each scenario.

Finally, the last part of this thesis develops a novel steganography method to be used for SOAP messages within Web services environments. This method is based on changing the order of XML elements according to a secret message. This method has a high imperceptibility; it leaves almost no trail because it uses the communication protocol as a cover medium, and keeps the structure and size of the SOAP message intact. The method is empirically validated using a feasible scenario so as to indicate its utility and value.

ACKNOWLEDGEMENTS

I would not have been able to complete this thesis without the aid and support of the kind people around me. First and foremost, I am grateful to my supervisor, Dr George Ghinea, who has offered me invaluable support and guidance throughout my Ph.D. with his knowledge and patience.

I owe my sincerest gratitude to my wife, Hala, and our precious children, Dana and Waleed, for their understanding and endless love at all times. My parents, brother and sister have given me their continuous support throughout, for which my mere expression of thanks does not suffice.

Last, but by no means least, I would also like to convey thanks to all my fellow colleagues, the academic and support staff in the Department of Information Systems and Computing at Brunel University. I am grateful to all of those with whom I have had the pleasure to work during the course of my Ph.D.

DECLARATION

The following papers have been published (or submitted for publication) as a direct result of the research discussed in this thesis:

Alrouh, B., Almohammad, A. & Ghinea, G. (2011) 'Information Hiding in SOAP Messages: A Steganographic Method for Web Services', *International Journal for Information Security Research (IJISR),* 1 (1), pp. 61-70.

Alrouh, B., Almohammad, A. & Ghinea, G. (2010) 'SOAP Message-based Steganography', *Proceedings of the 5th International Conference for Internet Technology and Secured Transactions (ICITST-2010), London, UK, 8-11 November 2010,* IEEE.

Alrouh, B., Al-Debei, M.M. & Ghinea, G. (2010) 'Developing a Decision-Making Framework for Web Service Security Profiles: A Design-Science Paradigm', *Proceedings of the 1st International Conference on Intelligent Semantic Web-Services and Applications, Amman, Jordan,14-16 June 2010,* New York, USA: ACM, pp. 28:1-28:7.

AbouTrab, M., Alrouh, B., Counsell, S., Hierons, R. & Ghinea, G. (2010) 'A Multi-criteria Decision Making Framework for Real Time Model-Based Testing' in *Testing – Practice and Research Techniques,* eds. L. Bottaci & G. Fraser, Springer Berlin / Heidelberg, pp. 194-197.

Alrouh, B. & Ghinea, G. (2009) 'A Performance Evaluation of Security Mechanisms for Web Services', *Proceedings of the Fifth International Conference on Information Assurance and Security (IAS '09), Xi'an, China, 18-20 August 2009,* IEEE CPS, pp. 715-718.

Alrouh, B., Zineddin, B. & Ghinea, G. 'Selecting Web Services Security Profiles: A Multi-Criteria Decision Making Approach', Submitted to: *International Journal of Web Services Research (JWSR)*.

AbouTrab, M.S., Alrouh, B., Counsell, S., Hierons, R.M. & Ghinea, G. 'Prioritizing Timed Automata Based Test Sets: A Multi-Criteria Decision Approach', Submitted to: *Journal of Information and Software Technology (IST)*.

ABBREVIATIONS

AHP	Analytical Hierarchy Process
ANP	Analytical Network Process
COA	Component-Oriented Architecture
COM/DCOM	Component Object Model/ Distributed Component Object Model
CORBA	Common Object Request Broker Architecture
CPU	CPU
CR: CI/RI	Consistency Ratio
DEA	Data Envelopment Analysis
DMU	Decision Making Units
Dsig	Digital Signature
DSR	Design-Science Research
Enc	Encryption
End-Cert	Endorsing Certificate
FA	Factor Analysis dimension reduction method
FTP	File Transfer Protocol
HOK	SAML Holder of Key
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPs	Secure HTTP
IETF	The Internet Engineering Task Force
J2EE	Java 2 Platform, Enterprise Edition
JAX-WS	The Java API for XML Web Services
Kerb	Symmetric Binding with Kerberos Tokens
K-T Decision Analysis	Kepner-Tregoe Decision Analysis
MA	Message Authentication over SSL
MAUT	Multi-Attribute Utility Theory
MCDM	Multi-Criteria Decision Making

MCS	Mutual Certificates Security
OASIS	The Organization for the Advancement of Structured Information Standards
OOA	Object-Oriented Architecture
PKI	Public Key Infrastructure
QoS	Quality of Service
RA	Ranking Approach
REST	REpresentational State Transfer
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RTT	Round Trip Time
RTTIP	Round Trip Time Increment Percentage
SA	SAML Authorisation over SSL
SAML	Security Assertion Markup Language
SMAR	The Simple Multi-Attribute Rating technique
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
SSO	Single Sign-On
STS	Security Token Service
STS-End	*STS Issued Token Endorsing Token*
STS-SC	STS Issued Token with Service Certificate
SV	SAML Sender Vouches with Certificates
UA	Username Authentication with Symmetric Key
UDDI	Universal Description, Discovery and Integration
UDP	Username with digest passwords
URI	Uniform Resource Identifier
VPN	Virtual Private Networks
W3C	The World Wide Web Consortium

WS- SecurityPolicy	Web Services Security Policy
WSDL	Web Services Description Language
WSIT	Web Services Interoperability Technologies
WS-Policy	Web Services Policy
WS-ReliableMessaging	Web Services Reliable Messaging
WS-SecureConversation	Web Services Secure Conversation
WS-Security (WSS)	Web Services Security
WS-Trust	Web Services Trust
XACML	eXtensible Access Control Markup Language
X-KISS	XML Key Information Service Specification
XKMS	XML Key Management Specification
X-KRSS	XML Key Registration Service Specification
XML	eXtensible Markup Language
XrML	Extensible Rights Markup Language

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1: Introduction

1.1 Overview

The Internet has changed the business world in a revolutionary way to a virtual world where the customer is served around the clock and around the world. In order to increase the productivity, decrease the cost and satisfy the customer, most of the organisations have shifted their business policies from a traditional way into applying Internet-based technologies (Yang, 2002; Rao et al., 2004). In this context, distributed computing has been the magical key enabling the business over the Internet (Nagappan, Skoczylas & Sriganesh, 2003).

In early stages, two basic standards, HyperText Markup Language (HTML) and HyperText Transfer Protocol (HTTP), were developed in order to enable the sharing of documents across the distributed network which led to the great success of the Web. Later, the Web became the preferable platform for many applications especially e-commerce, which illustrated the need of transferring the Web from the human-centric paradigm to the application-centric paradigm (Cerami, 2002; Hondo, Nagaratnam & Nadalin, 2002; Nandigam, Gudivada & Kalavala, 2005).

Many technologies have been developed to enable the movement from the human end-user interaction to the application-application interaction. Each of them has succeeded in its mission to a certain degree, but most of these systems consisted of ad hoc solutions (Cerami, 2002). According to Nandigam, Gudivada & Kalavala (2005), many solutions, such as Remote Procedure Call (RPC), Common Object Request Broker Architecture (CORBA), Component Object Model/ Distributed Component Object Model (COM/DCOM), Java Remote Method Invocation (RMI) and Message based application integration, had the idea of component-based software. However, they were determined by difficulties that appeared because of the non-transparency and dependency on programming languages, operating systems, data representations and network protocols.

The idea of Web services came about to overcome the previously mentioned problems, while offering the promise of automated Web with some standardisation in order to lower the barrier of application integration (Cerami, 2002).

A Web service is an application that is available on the Internet, or intranet, uses a standardised eXtensible Markup Language (XML) messaging system, and independent of any programming language or operating system (Cerami, 2002; Nakamur, Hada & Neyama, 2002; Geer, 2003). This technology is based on the standard Internet protocols, such as HTTP, File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP), as well as XML-based protocols, such as the Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), Universal Description, Discovery and Integration (UDDI) and Web Services Security (WS-Security) (Yang, 2002; Rao et al., 2004; Hondo, Nagaratnam & Nadalin, 2002). Web services provide a standardised way for applications to expose their functionality and communicate with other applications over a network, regardless of its implementation, programming language or platform (Singh et al., 2004).

The idea of applications from different parties communicating together raises a security threat, however. The security of exchanged messages is thus an important issue to be taken into consideration in Web services. The recipient of the message should be able to confirm its integrity and assure that it has not been modified. Additionally, the message should be delivered to the recipient confidentially where only the authorised users could read it, know the identity of the sender and determine the operation requested in the message (Mi et al., 2005). The challenge of Web services security is to understand and consider the risks of securing a Web-based service depending on the existing security techniques and simultaneously follow evolving standards in order to bridge the gap in Web services security. Any security model should illustrate the data flow through an application and network topology without exposing it to undue risk (Hondo, Nagaratnam & Nadalin, 2002).

1.2 Research Motivations

As the interaction between service providers and requesters occurs via XML-based SOAP messages, securing Web services tends to make these messages longer than they would be otherwise and consequently require interpretation by XML parsers on both sides, which reduces the performance of Web services (Menasce, 2002). The cost, in terms of performance, of securing Web services can be significant. The trade-off between performance and security depends primarily on the security approach that is used to secure the Web service (Novakouski et al., 2010). There are occasions when a simple transport-level security protocol, such as the Secure Sockets Layer (SSL) standard, is sufficient to meet the security requirements. However, for many types of applications, SSL alone is insufficient and more rigorous message-based mechanisms should be implemented (Sosnoski, 2009).

Moreover, as different standards and techniques for securing Web services have been developed by several organisations and various members of industry, some of these standards and techniques complement and extend each other, while others conflict or compete in the mix. Thus, selecting an appropriate security profile represents a complex dilemma for Web services architects and developers. This is due to the following three reasons: (1) There is no one supreme Web service security profile for all cases; i.e. different Web services security profiles achieve different requirements; (2) Applying different security profiles normally results in different quality measurements of a certain Web service; and (3) Different organisations usually have different requirements and even the same organisation may establish different requirements across different software applications.

Furthermore, choosing the best-suited Web service profile to be deployed for a certain application in a particular organisation is a complex undertaking decision-making task. This decision is usually uninformed as in many cases the decision is based on the sole experience of the developer. In some other cases, developers tend to use a profile that is (a) configurable based on the organisation's infrastructure; (b) developers are familiar with; or (c) supported by existing providers, without taking into consideration other security profiles. This is in fact represents the problem space.

Accordingly, the selection of a candidate profile, in an ideal world, should depend on evaluating the available security profiles against all the requirements of the given Web services application. Ideally, Web services developers specify a set of requirements according to the service provider's security preferences and technological constraints, taking into consideration the level of performance expected by the service consumer. The requirements can be divided into several dimensions according to different independent criteria and sub-criteria to create a multi-dimensional model. Each criterion and sub-criterion is given a weight according to its level of importance in the application. Each candidate solution is also weighted against each sub-criterion according to what extent this particular solution matches the requirement represented by this sub-criterion.

Generally, service providers set these weights on the basis of judgment. When there are a small number of selection criteria, direct judgments are possible; yet when the number of requirements is large, the direct judgement may favour the key elements only (Godse, Sonar & Mulik, 2008). In such a multi-criteria decision making model, it is important to have quantifiable values that are rational and consistent.

1.3 Research Aim and Objectives

The previous section (1.2) has highlighted that there is a considerable number of different security profiles and mechanisms that can be employed for Web services. Selecting a security profile for a given Web service often requires understanding the trade-offs between security and performance. Accordingly, the main aim of this research is thus:

> *To develop a prototypical framework that aids Web services architects to select the best suited security approach that satisfies the security and performance requirements of a given Web services application.*

In fulfilling this aim, the following objectives are considered important to be achieved:

Objective 1: Design a performance testing model to understand the cost of applying security profiles on the performance of Web services.

Objective 2: Develop a decision making framework based on the results gathered from the testing model, as well as the security requirements and system limitations.

Objective 3: Evaluate the developed framework through the use of real scenarios so as to indicate the framework's utility and value.

Objective 4: Explore and evaluate the feasibility of using steganography as an alternative approach to secure Web services.

1.4 Research Approach

The paradigm followed in this research is the Design-Science Research (DSR). This research aims, by utilising DSR, at producing an artefact in the form of a framework to help architects and developers to choose the best-suited security approaches for Web services applications. The aim of this research is highly consistent with the general aim of DSR.

In the context of this research, designing and developing a framework for selecting the most fitting security profile for a certain Web services application is the tackled wicked problem. In the context of design-science research, the term "wicked problems" can be described as unstructured decision-making activities and settings. This is because these types of decisions are normally "poorly formulated, confusing, and permeated with conflicting values of many decision makers or other stakeholders" (Pries-Heje & Baskerville, 2008).

Design artefacts are classified by March & Smith (1995), and anchored by Hevner et al. (2004), into *constructs* of vocabulary and symbols, *models* representing reality with appropriate levels of abstraction, *methods* in the form of algorithms and practices, and *instantiations* which are implemented systems and/or their prototypes developed as proof-of-concepts (Al-Debei & Fitzgerald, 2010). The developed framework in this thesis represents a *model* artefact which includes *constructs*. While its application and use in the course of the Analytical Hierarchy Process and the steganographical embedding/extracting algorithms represents the *method* artefacts. Developing the AHP and Steganography applications as software as well as the usage of real scenarios to validate them denote the *instantiations*.

The scheme to construct design artefacts is still very broad. Two main and general processes are identified by March & Smith (1995) as *build* and *evaluate*. Whilst building design artefacts demonstrate feasibility, they are evaluated against criteria of value to a community of intended users to ensure utility, quality, and efficacy (Hevner et al., 2004). Importantly, Design Science Research stresses the importance of *iterations* in producing the design artefacts and assumes that reality and knowledge emerge throughout the iterations effort (Markus, Majchrzak & Gasser, 2002; Vaishnavi & Kuechler, 2009).

1.5 Thesis Structure

The rest of this thesis is structured as follows:

Chapter 2 provides a literature review on Web services security and its related specifications. The trade-off between security and performance is highlighted and several approaches to analyse the performance of Web services security and selecting the best security approach are explored. The discussion identifies the research gaps that this thesis is addressing.

Chapter 3 details the research methodology employed in this thesis. A theoretical grounding of Design Science Research (DSR) is provided in this chapter. Thereafter, The DSR paradigm is justified as a suitable approach for this research. The research conducted in this thesis is then explained in line with the DSR research cycle.

Chapter 4 describes a set of performance tests conducted to evaluate the performance of various security profiles applied on a Web service, and tested with different initial message sizes. The results are used to compare the performance of SSL-based profiles against the message-level security ones. We also analyse the penalty of using SAML-based, STS-based, symmetric vs. asymmetric and reliability profiles on the performance of Web services.

Chapter 5 introduces a novel Analytical Hierarchy Process (AHP) decision-making framework for Web service security profiles. This framework represents the solution space which aims to make the decision in this context more informed, systematic, manageable, and effective by providing developers with an approach through which they can prioritise the security requirements, rank the available alternatives accordingly, and then select the profile that best fulfils their defined requirements. Three real-life scenarios are then tested using this framework and the results are compared to those presented in documentations of best practices to verify its efficiency.

Chapter 6 discusses a number of steganographic studies in text, XML as well as SOAP messages, and then proposes a novel steganography method to be used for SOAP messages within Web services environments. The method is based on rearranging the order of specific XML elements according to a secret message. This method is then empirically validated using a feasible example so as to illustrate its utility and value.

Chapter 7 summarises the research findings and outlines the research contributions to the knowledge. Finally, the limitations of this research are discussed and directions for future research are explored.

Chapter 2: Web Services Security

2.1 Overview

This chapter provides an overview of the Web services technology and its security standards and specifications. It also reviews previous research in the fields of performance analysis and selecting the most suitable approach to secure Web services. Accordingly, the chapter is organised as follows: the Web services technology is presented in section 2.2. The Service Oriented Architecture (SOA) is described in section 2.3. Section 2.4 explores the different Web service standards. The benefits of Web services and the challenges that face their development are mentioned in sections 2.5 and 2.6, respectively. Section 2.7 defines the security requirements of Web services and Section 2.8 describes the basic cryptographic concepts. Various XML and Web services security standards are explored in section 2.9, while the Web Services Interoperability Technology (WSIT) is introduced in section 2.10. Related studies in the subjects of performance analysis and decision making are reviewed in sections 2.11 and 2.12. Finally, section 2.13 summarises the chapter and highlights the research gap.

2.2 Web Services

It is generally accepted that Web services are applications that are available on the Internet or the intranet, use a standardised XML messaging system, and are independent of any programming language or operating system (Cerami, 2002; Nakamur, Hada & Neyama, 2002; Geer, 2003). This technology is based on the standard Internet protocols (e.g., HTTP, FTP, and SMTP) as well as XML-based protocols (such as SOAP, WSDL, UDDI and WSS) (Yang, 2002; Rao et al., 2004; Hondo, Nagaratnam & Nadalin, 2002). The World Wide Web Consortium (W3C) defines a Web service as “a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialisation in conjunction with other Web-related standards." (Booth et al., 2004).

Web services' applications can be implemented with different programming models. These models fall into three main categories (Goncalves, 2010), which shall now be presented:

2.2.1 XML-RPC Web Services

XML-RPC is a remote procedure call protocol. This protocol uses XML for marking up the Web services requests and responses (Lerner, 2001; Zimmerman, 2007). In an XML-RPC model, the service requester passes a request which contains the method name and parameters wrapped in XML that defines their data types. The response comes back with similar data (Zimmerman, 2007). RPC-based Web services are easy to implement when using scripting languages, which have very loose data types (Muller, 2010). However, the main criticism of XML-RPC is not being loosely-coupled, because it is usually implemented by mapping services directly to language-specific functions or method calls (Muller, 2010). Moreover, this mechanism does not handle advanced data structures (Lerner, 2001). Therefore, many vendors felt this approach to be a cul-de-sac and stopped their support for RPC-style Web services.

2.2.2 REST-based Web Services

REST (REpresentational State Transfer) Web services are collections of Web resources, which are identified by their Uniform Resource Identifiers (URIs). Every document and process is modelled as a Web resource with a unique URI (Goncalves, 2010). These Web resources are manipulated by actions that can be specified in an HTTP header. Messages can be exchanged in any format, such as

XML, HTML, without the need for the WSDL or SOAP protocols. A Web browser can serve as a client in many cases. A Web service interacts with resources, rather than working with messages and operations. HTTP is the main protocol in REST, where only four methods are available: GET, PUT, POST, and DELETE (Goncalves, 2010; Netbeans, 2011). REST can typically be used for simpler applications, where HTTP is the appropriate protocol, and when the HTTP infrastructure alone can satisfy the security requirements of these applications (Netbeans, 2011).

2.2.3 SOAP-Based Web Services

The business logic of SOAP-based Web service is exposed on the net via a WSDL document. Messages are exchanged between the Web service and its clients using SOAP messages. SOAP-based Web services are suitable for complex applications, where complicated operations are required, or for applications requiring advanced security (Goncalves, 2010).

For the purpose of this thesis, and because the main focus is the security of Web services, all the discussion will be based on SOAP-based Web services.

2.3 Service-Oriented Architecture (SOA)

The word *services* in *Web services* refers to the Service-Oriented Architecture (SOA) (O’Neill et al., 2003). SOA is a recent development in distributed computing, in which an application can call a functionality from another application over a network (O’Neill et al., 2003). This architectural style supports software reusability by creating reusable services in comparison with the traditional Object-Oriented Architecture (OOA), which supports reusability by reusing classes or objects (Booch, 1986). The major difficulty with OOA is that objects are often too fine-grained for efficient reusability. Therefore, a Component-Oriented Architecture (COA) has emerged to use software components as reusable entities, which consist of a set of related classes, resources and configuration information (Singh et al., 2004). Although COA is a powerful way to design software systems, it does not address the additional issues emerging from current-day enterprise environments, which have become rather complicated because of using different software and hardware platforms, Internet-based distributed communication and enterprise application integration (Brown, Johnston & Kelly, 2002; Singh et al., 2004). The Service-Oriented Architecture addresses these issues by using services as reusable entities (He, 2003).

According to Cerami (2002), He (2003) and Carminati, Ferrari & Hung (2005), there are three major roles in the Service-Oriented Architecture:

- A service provider that implements the service and makes it available for a particular business purpose.
- A service requester that utilises an existing Web service to meet business requirements.
- A service registry which is a centralised place where developers can provide new Web services or use the existing ones.

An important aim of Service-Oriented Architecture is to achieve loose coupling among interacting software agents (He, 2003). To enable this, the following mechanisms should be available (Singh et al., 2004):

- A mechanism for clients to access services and registries.
- A mechanism for services to be registered with registries and for clients to search these registries for their required services. The Web services' architecture enables services that are located over the network with transparent locations to be dynamically discovered by clients.
- A mechanism for well-defined Web services' interfaces to be exposed in a way that enables clients to access those interfaces.

In a Web service model, functionality is published on a network where two important capabilities are provided: the first is the ability to find the functionality (discovery), and the second is the ability to connect to the functionality (binding). These capabilities are represented by three roles: Web services provider, Web services requester and Web services broker (O'Neill et al., 2003; Carminati, Ferrari & Hung, 2005). The publish-find-bind model is illustrated in Figure 2-1.

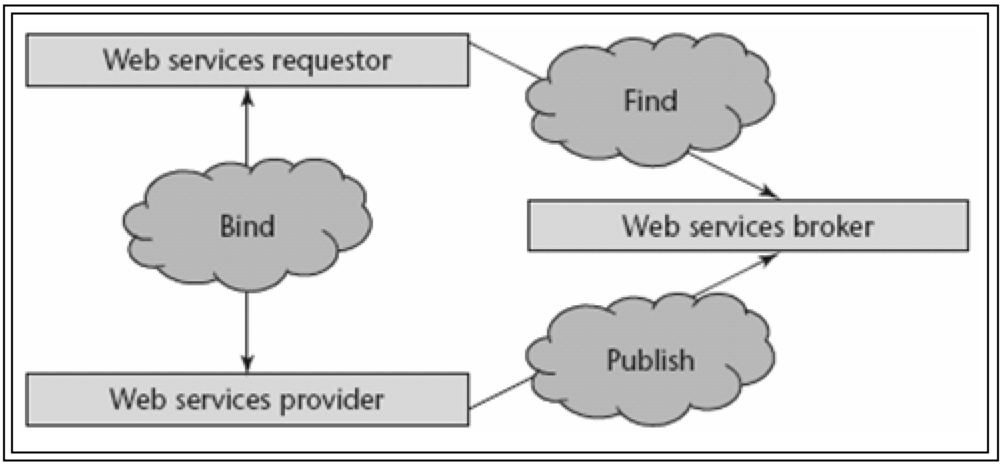

Figure 2-1: The Publish-Find-Bind Model (O'Neill et al., 2003)

2.4 Web Service Standards

Singh et al. (2004) have stated that for any technology to be successful, the technology standards must be widely accepted. To enable such a wide acceptance, Web services standards and systems that implement those standards should have the following criteria:

- "A Web service should be able to service requests from any client regardless of the platform on which the client is implemented".
- "A client should be able to find and use any Web service regardless of the service's implementation details or the platform on which it runs".

Standards enable Web services to achieve wide acceptance and interoperability by establishing a base of commonality.

The main Web services standards cover the following areas:

- Common markup language for communication.
- Common message format for exchanging information.
- Common service specification formats.
- Common means for service lookup.

Figure 2-2 illustrates a stack of the main standards on which Web services are generally based on:

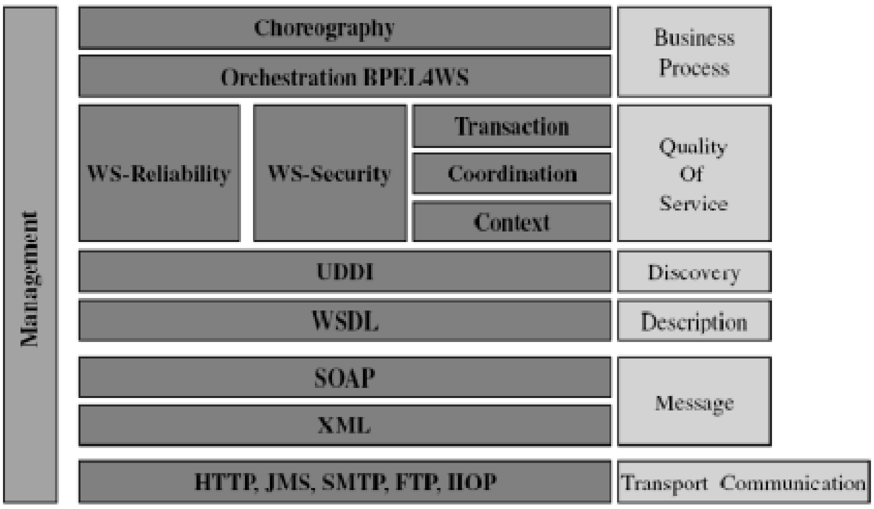

Figure 2-2: The Web Services Technology Stack (Albreshne, Fuhrer & Pasquier, 2009)

2.4.1 Extensible Markup Language (XML)

The eXtensible Markup Language (XML) is a simple and flexible text based markup language. This standard is accepted throughout the computer industry as it facilitates the communication between service providers and requesters using a common language. XML is also independent of any platform or technology. Messages in XML can be exchanged over the Internet using standard Internet protocols such as HTTP (Guruge, 2003; Siddiqui, 2003a).

Tags enclosed in angled brackets are used to mark XML data; the tags contain the meaning of the data they mark. The XML tag usage is different from HTML, which is oriented to displaying data. Thus, unlike HTML, display is not intrinsic in XML (Cerami, 2002).

XML is a product of the W3C. Therefore, changes to it will be supported by all leading parties. This means that when XML evolves, Web services can also evolve without having concerns about compatibility (Singh et al., 2004).

2.4.2 Simple Object Access Protocol (SOAP)

As XML fills the need for a common language, the Simple Object Access Protocol (SOAP) solves the need for a common messaging format (Mitra & Lafon, 2007). SOAP enables different objects to communicate by exchanging messages. SOAP is an XML-based protocol that uses data encoding format and HTTP/SMTP to transfer messages. SOAP does not require any specific technology at its endpoints because of its independency of programming languages and platforms. Moreover, SOAP is also supported by leading industrial parties (Singh et al., 2004).

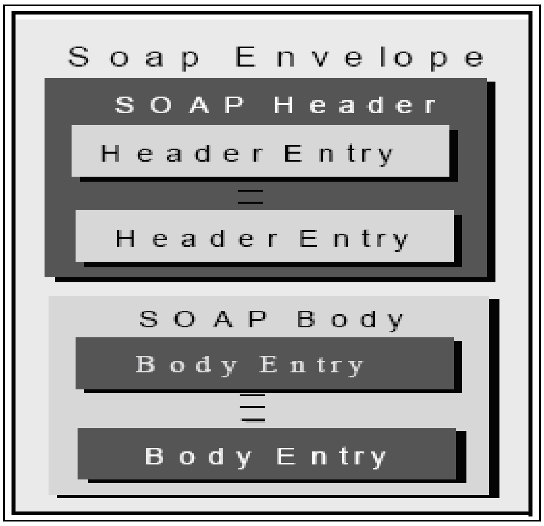

Figure 2-3: SOAP Message Structure (Singh et al., 2004)

SOAP defines an envelope, which contains a SOAP body, where the message is included, and an optional header. The SOAP envelope, body plus header, is an XML document (Mitra & Lafon, 2007). Figure 2-3 illustrates a SOAP envelope:

2.4.3 Web Services Description Language (WSDL)

The role of the Web Services Description Language (WSDL) is to define a standard way for specifying the details of a Web service (Christensen et al., 2001). Details of Web service interfaces, bindings and other deployment details are specified using a general-purpose XML schema. That enables clients without prior knowledge of the service to use that Web service (Adams & Boeyen, 2002; Singh et al., 2004).

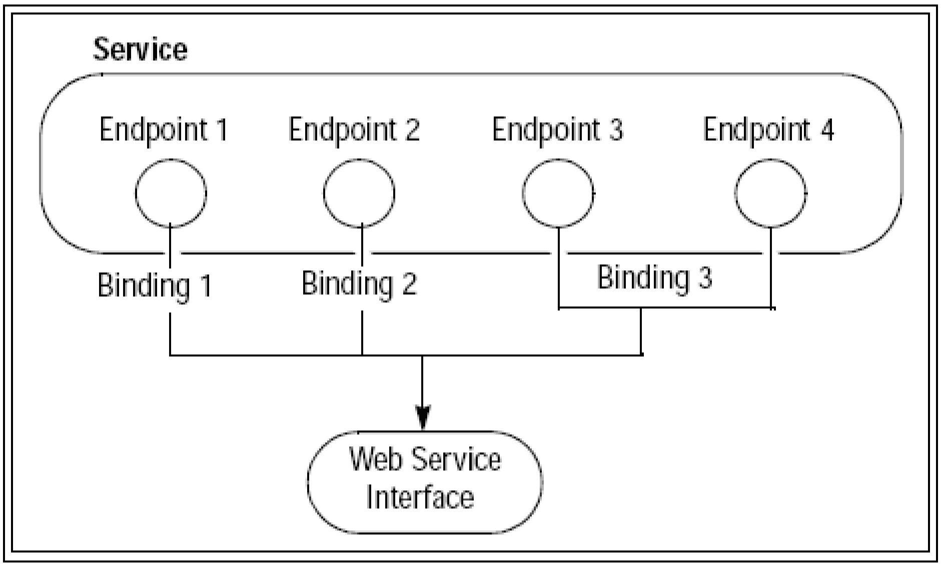

Figure 2-4: WSDL Service Description (Singh et al., 2004)

WSDL grammar describes Web services as a collection of communication endpoints, as shown in Figure 2-4. The exchanged data are specified as part of messages. Every action allowed at an endpoint is an operation. In addition, port types are grouped together collections of operations that are possible on an endpoint. The port types, operations as well as messages are all abstract definitions, which do not hold deployment-dependent details in order to enable their reuse. A binding is specified by a protocol and data format specifications for a particular port type. A port is defined when a network address is associated with a reusable binding. A collection of ports, in turn, defines a service. Furthermore, WSDL specifies a common binding mechanism to bring together all protocols and data formats with an abstract message, operation, or endpoint (Singh et al., 2004).

2.4.4 Universal Description, Discovery, and Integration (UDDI)

The Universal Description, Discovery, and Integration (UDDI) specification defines a standard way for registering, deregistering, and looking up Web services (Clement et al., 2004). UDDI is a standards-based specification for Web service registration, description, and discovery.

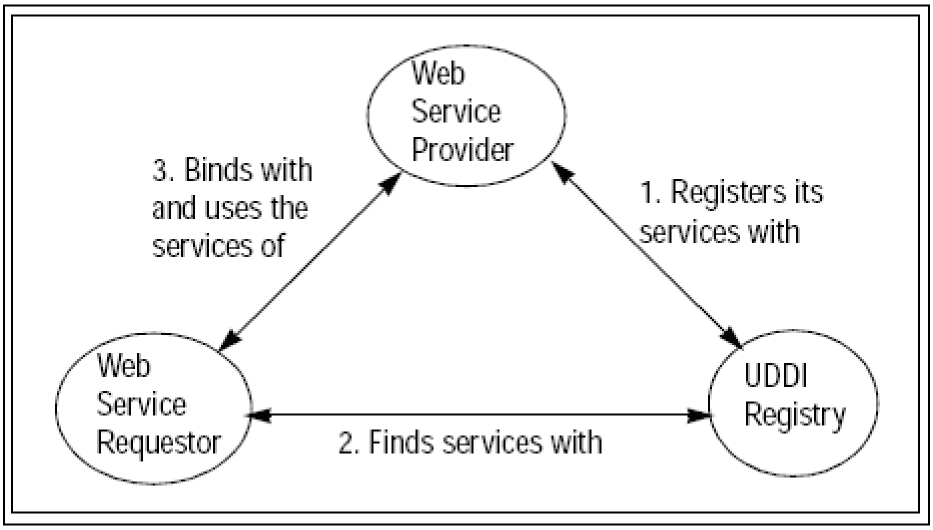

Figure 2-5: Role of Registry in a Web Service (Singh et al., 2004)

The main purpose of UDDI registry is to enable service providers to register their services with a "broker" and requesters to find services advertised by this broker. The role that UDDI plays between the service requester and the service provider ends when a requester finds the service; a service provider registers its services with the broker (i.e. UDDI registry). A service requester then searches for the required service in the UDDI registry. When the required service is found, the service requester binds directly with the service provider to use that service (Singh et al., 2004), as illustrated in Figure 2-5.

Using the UDDI specification, an XML schema is defined for applications wanting to use the registry. A service provider registering its Web service with a UDDI registry must provide service, business, binding and technical information about the service (Adams & Boeyen, 2002; Singh et al., 2004). This information is stored in a common format that contains three parts:

1. White pages that describe general business information such as name, description, phone numbers, etc.

2. Yellow pages that describe the business using terms of standard classifications (taxonomies), which follow standard industrial categorisations in order to enable locating services by industry, category, or geographical location information.

3. Green pages that list the service, binding, and service-specific technical information.

Web services standards have been widely accepted throughout enterprises and the popularity of Web services is increasing because of the benefits they provide.

2.5 Benefits of Web Services

According to Singh et al. (2004) and Albreshne, Fuhrer & Pasquier (2009), these are the benefits enterprises gain from adopting Web services:

2.5.1 Interoperability

The key feature of the Web services model is that it allows various distributed services to be implemented using different programming languages and executed on a variety of software platforms and architectures. This is a vital benefit in large enterprises, where different solutions and systems that often require different platforms are developed over the time; Web services thus enable interoperability in these heterogeneous environments and various systems can use the services to easily interoperate with each other.

2.5.2 Expanding Business Services Through the Web

In the business world, Web services can be used to build upon the benefits of the World Wide Web for its operations. For example, product catalogues can be made available to retailers through a Web service to achieve better supply chain management.

2.5.3 Enabling Integration With Existing Systems

Most enterprises have a huge amount of data stored in existing enterprise information systems and databases, and the cost of replacing these systems may not be a reasonable option. Web services provide developers with standard ways to access middle-tier and back-end services without forcing developers to learn new programming models or styles.

2.5.4 Freedom of Choice

Web service standards have expanded the marketplace for tools, products, and technologies. This gives organisations an extensive diversity of choices, and they can select configurations that best cover their application requirements. Developers can improve their productivity because they can choose from a ready market of pre-built application components rather than having to develop their own solutions. Moreover, these tools enable developers to move quickly and easily from one configuration to another as needed. Web services also guarantee the standardisation of tools, so that developers can choose to adopt tools from either server vendors or a third party, depending on the requirements.

2.5.5 Supporting More Client Types

Web services enable more client types to use applications and services. They aim at supporting interoperability, and therefore, extending the reach of existing applications or services to various client types. This does not depend on the platform on which the client is based. For example, a client can be based on a Java or Microsoft platform or even a wireless platform.

2.5.6 Increasing Programming Productivity

Productivity in an information-driven economy demands the ability of developing and deploying applications in a reasonable time frame. It means that applications should go rapidly from the prototyping stage to the production phase and simultaneously continue to evolve even after they are placed into production. Productivity is improved when application development teams have standard ways to access the services required by multi-tier applications and standard means to support a different types of clients, and that what Web services provide by creating a common programming standard. Before the appearance of Web services, distributed computing environments have had many different technologies, which are not always compatible. Developers have tried to tie several different systems together, such as custom and standard database management systems and transaction processors, with traditional Web technologies, but have had to deal with a large number of different programming models. Since Web services introduce a common standard across the Web, vendors, in order to stay in the competition, are more likely to develop better tools and technologies to attract developer, and so the whole industry benefits.

Thanks to its advantages, Web services technology has rapidly extended through enterprises. However, there are many challenges that face Web services technology. To benefit from this new technology, these challenges should be well addressed.

2.6 Challenges of Web Services Development

The Web services specifications are evolving quickly and sometimes in unexpected directions. These specifications vary in their degrees of maturity and therefore these technologies may change as they are extended to provide improved Web services support (Singh et al., 2004). Moreover, they are supported and maintained by various standards organisations, and as a result they may complement, overlap or compete with each other (Lakshminarayanan, 2010); thus, it is important to take these factors into consideration when developing Web services.

2.6.1 Security

Unsurprisingly, security is as important for Web services as it is for other enterprise applications. As a matter of fact, security becomes an even more important aspect as applications on the Web expose their enterprise's processes and business data to distributed, and not necessarily trusted, clients (Hondo, Nagaratnam & Nadalin, 2002; Holgersson & Soderstrom, 2005; Mahmoud, 2005).

2.6.2 Advancing Technologies

Although the basic Web services' standards and protocols (including WSDL and SOAP) have matured during the last decade, there is a growing number of supporting protocols , referred to as WS-* protocols , that have yet to reach the level or wide adoption SOAP and WSDL have achieved (Kontogiannis, 2008).

In addition, Interoperability is a persistent challenge, which Web services have already succeeded to realise, but achieving interoperability requires developing further standards to guarantee an interoperability degree that matches the vision of Web services (Lakshminarayanan, 2010).

Another challenge that faces Web services is the organising of multiple services for processing business logic (Singh et al., 2004). Often, a single business process is implemented as a sequence of stages in a workflow, where a separate service may be used to implement a particular stage in that workflow. Therefore, achieving the goal of a specific business logic process requires a high level of coordination between the services that compose this process (Singh et al., 2004).

2.6.3 Reliability

Reliability is the quality aspect that is concerned with how well a service is maintained and can be measured by the number of failures that occur during a specified period of time. In Web services' environments, achieving reliability is an obvious challenge due to the unreliable nature of the underlying protocols. For instance, the default HTTP best-effort delivery neither guarantees the delivery of all the sent packets nor delivering them in the same sending order (Singh et al., 2004; Wang et al., 2004).

2.6.4 Availability and Response Time

Availability is the probability that a Web service is present and ready to be used when required (Singh et al., 2004). One of the challenges that faces the developers of Web services is to maximise the availability of these services and minimise the amount of time a client has to wait before receiving a response from an invoked service.

2.6.5 Scalability

The scalability challenge concerns handling a large number of simultaneous client interactions. An efficient implementation of a Web services' system requires a good management of the system's resources and services, including database connections and XML parsing. Validating XML documents through XML parsing is an intensive process, when compared to its equivalent binary format. This can have a significant effect on the performance of Web services as it increases the payload size (Singh et al., 2004).

In this thesis, the focus will be on the security challenge. The crucial issue in Web services is how to ensure their security since they are based on exchanging messages through the Internet where there are always security risks regarding stealing, loosing or modifying these messages (Nakamur, Hada & Neyama, 2002). Geer (2003) has stated that Web services pose a difficulty to network administrators because Web services open up networks by enabling users from outside their networks to access databases, applications and internal users. Furthermore, Web services can perform large number of transactions in a short time which are difficult to be secured using traditional security techniques, such as Virtual Private Networks (VPNn) or Secure Socket Layer (SSL).

2.7 Security Requirements

There are some major security requirements that must be addressed to ensure the safety of exchanging Web services information through a network. These requirements are:

2.7.1 Confidentiality

In any networked system, the communicating parties need to exchange data while guaranteeing that only the expected receiver can read this data. This means that the exchanged data must be protected against eavesdroppers (Nakamur, Hada & Neyama, 2002; Geer, 2003). The term confidentiality in the field of information security refers to the requirement for exchanged data between two communicating parties not to be available to a third party that may try to pry into the communication (O'Neill et al., 2003). In order to achieve confidentiality, one approach is to use a private connection between the communicating parties, such as a dedicated line or a virtual private network (VPN). However, the critical information of a Web service, such as WSDL and SOAP messages, are usually exchanged through untrusted networks, the Internet most likely, where the private connection is not achievable and another approach is used to meet the requirement of confidentiality, that is encryption (Rao et al., 2004).

2.7.2 Integrity

The term integrity refers to the requirement of ensuring that transmitted information hasn't been changed or modified during transmission (Nakamur, Hada & Neyama, 2002; Geer, 2003). It does not mean preventing information from being tampered with since it is impossible in untrusted networks, especially the Internet. The next best thing is to detect this tampering if it occurs. The knowledge about the fact that tampering has occurred fulfils the integrity requirement. Integrity can be achieved using digital signature (O'Neill et al., 2003).

2.7.3 Non-repudiation

Non-repudiation literally means that the message originator cannot deny sending this message. Any doubt about the message sender throws confidentiality and integrity into question and the results could be either bad or disastrous. Digital signature and Public Key Infrastructure (PKI) technologies are used to deliver non-repudiation (Nakamur, Hada & Neyama, 2002; Geer, 2003).

2.7.4 Authentication

Authentication refers to establishing identity (Geer, 2003) which ensures that access to data and applications is limited to those who have the appropriate proof of identity (Nakamur, Hada & Neyama, 2002). As with standard Web traffic, the service provider should authenticate service requesters before sending Web services information (Yang, 2002). PKI technology could be also used for authentication. However, it is not the only available method for authentication. Biometrics, passwords and hardware devices, such as dongles and smart cards, are also used for authentication. They all are based on the idea of having a token in the possession of the entity that is authenticated and this token is either software-based or hardware-based. In the case of Web services the communication could be between software and another which adds a new twist on authentication. The scenario is a human user may authenticate directly to their systems using a human-machine authentication technique where they will not be running Web services directly. However, when a Web service starts, information about the authentication status must be carries in the Web service communication. This scenario is enabled by the Security Assertion Markup Language (SAML) (O'Neill et al., 2003).

2.7.5 Authorisation

Authorisation means determining the privileges of the user and deciding whether the entity is allowed to access particular resources and services or not (Yang, 2002; Nakamur, Hada & Neyama, 2002; Geer, 2003). Just because a user is authenticated does not mean that they are always authorised. Authorisation software allows an administrator to manage a policy for access control to services by giving different privileges to different users and groups. Single Sign-On (SSO) technologies, such as SAML, are used in Web services for both authentication and authorisation (O'Neill et al., 2003).

2.7.6 Availability

Availability could be an ambiguous security requirement but it is very essential. As well as Web services, security services themselves require availability. Otherwise they are meaningless because it is costly for any business if critical information is not available when needed (O'Neill et al., 2003).

2.7.7 Privacy

While confidentiality is the requirement that data that is in transit is not available to eavesdroppers, the privacy requirement concerns the privacy rights of the subject of the data. A strong encryption could protect the data while it is in transit. However, this data could not be well encrypted while storing it and if there are any back doors in the Web application or a direct connect to the database this data may be stolen. Privacy thus requires assuring that data is always protected (O’Neill et al., 2003).

2.8 Basic Cryptographic Concepts

In the world of Information Systems, cryptography is often seen as a synonymous with security (O’Neill et al., 2003) because security implies some core concepts , which are keys, cryptography, signature and certifications (Siddiqui, 2003b).

2.8.1 Cryptography

There are two encryption methods, asymmetric cryptography and symmetric cryptography. The first method uses a pair of keys (public key and private key). This pair is generated by a suitable cryptographic algorithm. The public key, as the name implies, is open for public use, while the private key should be kept confidential. When someone wants to send a confidential message to a specific party, the sender asks for the public key of the receiver and uses this key to encrypt the original message. The receiver uses the private key to decrypt the message. No one else will be able to decrypt the message since only the receiver has the private key. The other encryption method (symmetric cryptography) uses the same key for encryption and decryption which makes this method less expensive than the first one. Therefore, the asymmetric method is used only to exchange the shared secret. When both of the parties know the shared secret, they start using the symmetric encryption (O’Neill et al., 2003; Siddiqui, 2003b; Tatsubori, Imamura & Nakamura, 2004).

2.8.2 Message Digests

The message digest method introduces an added field (value) to the original message that occurs from applying a digest calculation on the message data. The sender sends the digest value with the message, while the receiver reapplies the same digest calculation and compares the resulted value with the digest value. If the message has been altered, these values will not match. The drawback of this method is that the change will not be detectable at the receiver side if both the message and its digest are altered during the transmission (Siddiqui, 2003b).

2.8.3 Digital Signature

In this method, the digest algorithm is used to generate the digest value of the message, and then the private key is produced to generate a digital signature over the digest value. The message receiver repeats the digest calculation and then uses the public key to verify the signature. This method assures that the message hasn’t been altered after applying the digest calculation (message integrity) and the message is surely coming from the owner of the public key (user authentication) (Yang, 2002; Siddiqui, 2003b).

2.8.4 Certification

The certificate consists mainly of two fields of information, the identification of the certificate owner and the public key of the certificate owner. The certificate issuing authority also signs the certificate using its own private key, which enables any interested party to check the integrity of the certificate by verifying the signature (Yang, 2002).

2.9 XML and Web Services Security Standards

With the aim of fulfilling the security requirements and providing a framework to secure XML-based applications, standards organisations, such as the W3C and the Organization for the Advancement of Structured Information Standards (OASIS), have presented various security specifications that coordinate with each other to form modules of *XML firewalls* (Siddiqui, 2003a). Figure 2-6 illustrates the most common XML and Web services standards and their dependencies.

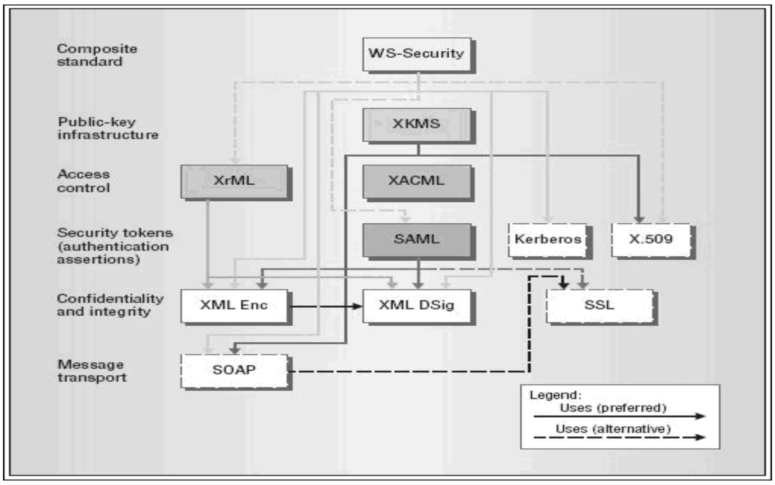

Figure 2-6: XML and WSS standards and their dependencies (Naedele, 2003)

The security specifications can be divided into three main categories (Nordbotten, 2009). These categories and their underlying specifications are:

2.9.1 XML Security

XML Digital Signature (XML DSig)

The XML Signature specification is a joint effort of the W3C and The Internet Engineering Task Force (IETF) aiming to provide data integrity, message authentication and signer authentication features wrapped in an XML format (Siddiqui, 2003a; Naedele, 2003; W3C, 2002b).

The XML signature specification describes XML syntax to associate between cryptographic signature and XML documents. It includes procedures for establishing and verifying XML signatures (Naedele, 2003). Digital signature of one party could be read by another because the machines work with the same encrypted digest for the same section of XML document (Siddiqui, 2003a).

XML Encryption (XML Enc)

W3C's XML Encryption specification uses encryption techniques to address the requirement of data confidentiality (Geer, 2003; Siddiqui, 2003a; Naedele, 2003; W3C, 2002a).

The XML encryption specification defines the process for encrypting and representing encrypted data in XML documents. All or just part of the XML could be encrypted in the message. Therefore, only information that is confidential is encrypted while the unconfident information could be sent unencrypted (Geer, 2003).

XML Key Management Specification (XKMS)

XKMS (W3C, 2001) defines a Web service interface that combines the interoperability of XML with the security of Public Key Infrastructure (PKI) in order to manage the security of PKI-based application. XKMS consists of two sub-protocols; XML Key Information Service Specification (X-KISS) and XML Key Registration Service Specification (X-KRSS) (Nordbotten, 2009). X-KISS is used for locating and retrieving public keys from a key server to perform the encryption or signature verification, while X-KRSS defines an interface that can register, revoke and recover escrowed keys from a key server (Naedele, 2003).

2.9.2 Security Markup Languages

eXtensible Access Control Markup Language (XACML)

Presented by OASIS, XACML enables developers to express their authorisation and access policies in XML (Geer, 2003; Siddiqui, 2003a; Naedele, 2003).

XACML specifications enable access control policies to be expressed in XML. It expresses sophisticated access control model and fine-grained access policies in XML documents, or any other e-resources. Using this specification, the access control policies control how XML documents appear to the end user (Nagappan, Skoczylas & Sriganesh, 2003; Naedele, 2003).

Extensible Rights Markup Language (XrML)

XrML is an easy-use general-use specification focuses on expressing rights and conditions related to digital services and resources (e.g., expiration times). Thus, XrML deals with digital rights management, but it is not suitable for complex access policy or rule sets, which are addressed by XACML. Both XACML and XrML don’t deal with authentication and protection directly; instead they leave these matters to the underlying encryption and digital signature protocols (Naedele, 2003).

Security Assertion Markup Language (SAML)

OASIS provides a possibility for partner applications to share user authentication and authorisation information, by achieving the single sign-on feature without using cookies as in the normal way (Geer, 2003; Siddiqui, 2003a; Naedele, 2003).

SAML is an XML-based framework for exchange security information. Such exchanges occur usually between interacting applications that do not always share the same authentication and authorisation techniques. However, SAML assumes trust between participants because SAML does not provide this trust by itself. It refers to XML Enc and XML DSig to establish this trust (Hondo, Nagaratnam & Nadalin, 2002; Geer, 2003; Naedele, 2003).

2.9.3 Web Services Security

Web Services Security (WS-Security)

OASIS defines this specification depending on The XML Signature and XML Encryption specifications in order to include integrity, confidentiality and single message authentication features within a SOAP message (Geer, 2003; Siddiqui, 2003a). WS-Security (Nadalin et al., 2006) specification provides a way for Web services developers to implement several different security models throughout attaching security data to the headers of SOAP messages (SOAP extensions) (Geer, 2003).

Web Services Policy (WS-Policy)

The WS-Policy (Bajaj et al., 2006) specification defines a framework for expressing policies that refer to interoperability capabilities and requirements in a Web services-based system (Vedamuthu et al., 2007).

Web Services Security Policy (WS- SecurityPolicy)

The WS-SecurityPolicy (Nadalin et al., 2007c) specification defines a standard set of assertions that represent security requirements and preferences for Web service endpoints in order to describe the preferable way of securing the communications path (Sun Microsystems Inc., 2010).

Web Services Trust (WS-Trust)

WS-Trust (Nadalin et al., 2007b) supplements the functionality of WS-Security and Web Services Policy by defining security tokens management mechanisms (such as issuing, renewing, cancelling, and validating security tokens). In a Web services security model, the service consumer may use WS-Trust, after discovering what security token is required, to obtain the required token from a Security Token Service (STS) (Vedamuthu et al., 2007).

Web Services Secure Conversation (WS-SecureConversation)

WS-SecureConversation (Nadalin et al., 2007a) extends WS-Security and WS-Trust to provide mechanisms for establishing and identifying a security context in order to support exchanging multiple messages. The communicating parties share the security context for the duration of the communication session. This can potentially improve the overall performance of subsequent message exchanges, because more efficient keys or new key material can be exchanged.

2.10 Web Services Interoperability Technologies (WSIT)

In order to improve Web services Quality of Service (QoS) and to enable interoperability between *Java* and *.Net* Web services, The Web Services Interoperability Technologies (WSIT) (Sun Microsystems Inc., 2010; Sun Microsystems Inc., 2007) has been developed as a joint effort between *Sun Microsystems, Inc.* (later merged with *Oracle USA, Inc.* to become *Oracle America, Inc.*) and *Microsoft Corporation* as part of the *Metro* Web services stack.

Enterprise features are supported in Metro's WSIT through the implementation of several open Web services standards and specifications, such as message optimisation, reliable messaging, and security. Figure 2-7 shows the underlying services that were implemented for each technology.

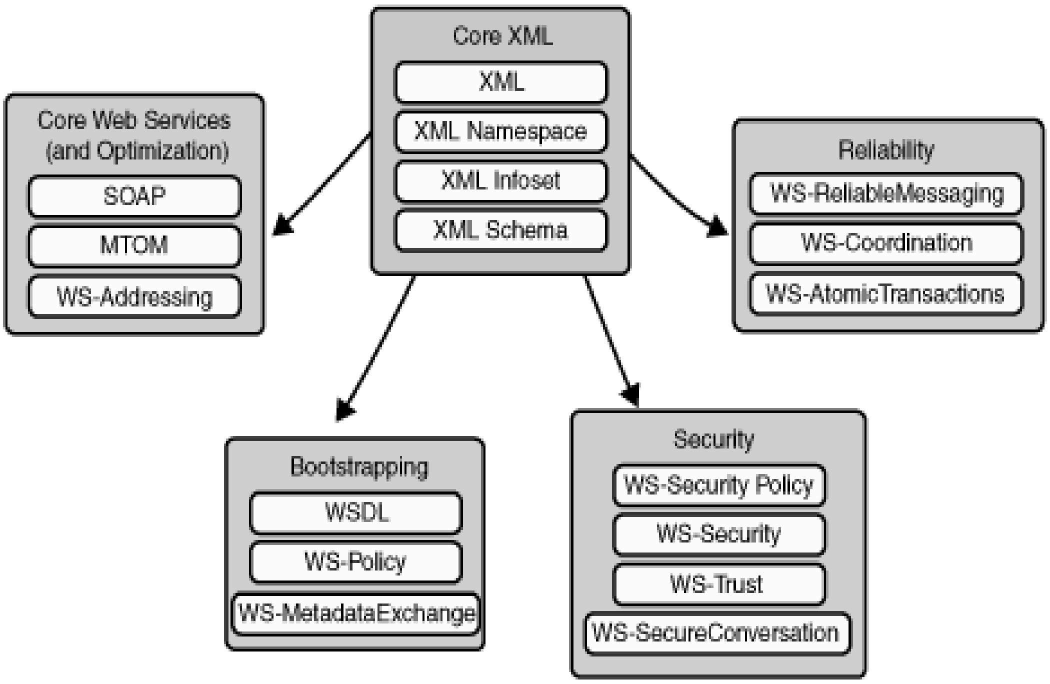

Figure 2-7: WSIT Web Services Features (Sun Microsystems Inc., 2007)

2.10.1 WSIT Reliable Messaging Technology

Reliability is the QoS aspect of a Web service representing how well it maintains its service quality. Reliability is often measured by the number of failures that occur in a given time period. Reliability in Web services systems may be more difficult to maintain because of the unreliable nature of the underlying transport specifications, such as HTTP (Sun Microsystems Inc., 2010).

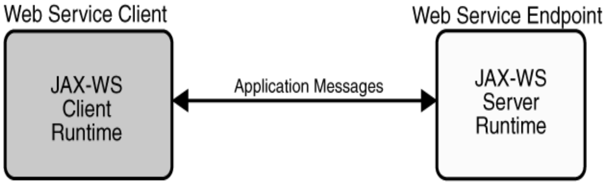

Figure 2-8: Application Message Exchange without Reliable Messaging

(Sun Microsystems Inc., 2007)

Without reliability, the receiving endpoints would not know if messages are lost, duplicated, or reordered as these messages transfer over the HTTP connection without delivery assurances (Figure 2-8).

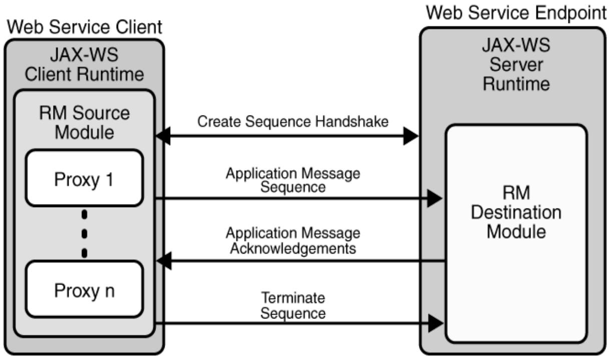

Figure 2-9: Application Message Exchange with Reliable Messaging Enabled

(Sun Microsystems Inc., 2007)

The Web Services Reliable Messaging technology (WS-ReliableMessaging) defines a protocol that implements a QoS contract between an application and its underlying messaging processor service. This contract consists of four actions:

submit, deliver, respond and *notify*, and covers the following delivery assurances: *at most once, at least once, exactly once* and *in order* (Bertino et al., 2010).

Securing SOAP messages alone does not prevent them from being lost in transit, or delivered out of order (Bertino et al., 2010). Reliability enables systems to overcome the failure of losing messages in transit or delivering them out of order. If a message is lost, the sender endpoint resends the message until its receipt is acknowledged by the receiving endpoint, as illustrated in Figure 2-9. If these messages are received out of order, the receiving endpoint can rearrange the messages into the correct order.

2.10.2 WSIT Security Technology

WSIT implements several Web services security specifications to provide interoperability and secure communication between Web services endpoints, as well as any intermediary nodes, as presented in Figure 2.10. Security, as provided by WSIT, is an addition to the existing transport-level security, which may still be used when point-to-point security is all that is needed. Besides the main XML and Web services specifications, security is implemented in WSIT by adopting the following specifications:

- Web Services Security (WS-Security).
- Web Services Policy (WS-Policy).
- Web Services Trust (WS-Trust).
- Web Services Secure Conversation (WS-SecureConversation).
- Web Services Security Policy (WS-SecurityPolicy)

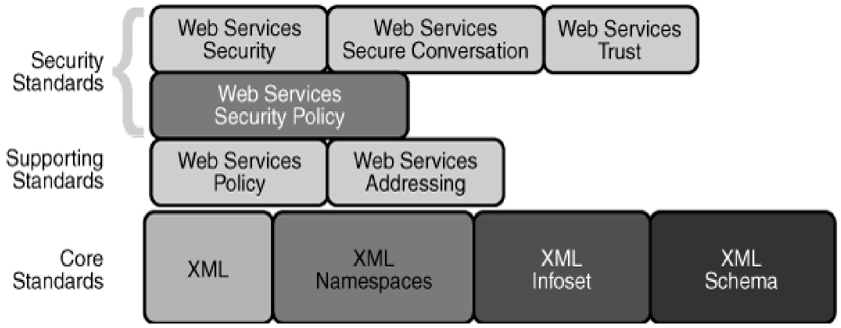

Figure 2.10: WSIT Security Specifications

2.10.3 WSIT Security Profiles

A security profile is a description of a secure communication exchange between a service provider and its client. Achieving different levels of security requires a combination of different security techniques (i.e. *security tokens*, *encryption*, *signature*, and even securing the entire wire using *SSL*). As a result, SOAP-messages can be safely delivered over the unsecured wire and can be processed, entirely or partially, by middleware services if required (Hatala, Eap & Shah, 2004).

WSIT provides a number of security profiles that can be applied to secure a Web service. Each profile represents a set of pre-defined security specifications and configurations. These security profiles are (Sun Microsystems Inc., 2010):

- *Username Authentication with Symmetric Key (UA)*: This profile depends on a symmetric key cryptography that is used for integrity and confidentiality. It relies on a single, shared secret key, generated at runtime and encrypted using the service's certificate. The client does not possess any certificates on its own, but instead sends its username/password for authentication.

- *Username with digest passwords (UDP)*: This profile is similar to UA, except that digest passwords are used in the username token and therefore is not required to be encrypted.

- *Mutual Certificates Security (MCS)*: This is an asymmetric cryptography profile that adds security via authentication and message protection that ensures integrity and confidentiality.

- *Symmetric Binding with Kerberos Tokens (Kerb)*: This profile authenticates the client using Kerberos Tokens. The integrity and confidentiality protection are established using symmetric keys, generated with the Kerberos Protocol.

- *Transport Security using SSL*: This profile uses Secure Sockets Layer (SSL) to protect the application during transport. Transport-layer security is provided by transport mechanisms used to transmit information over the wire between clients and providers, thus transport-layer security relies on HTTP Secure transport (HTTPS) using SSL. Transport security is a point-to-point security mechanism that can be used for authentication, message integrity, and confidentiality.

- *Message Authentication over SSL (MA)*: This profile attaches an authentication token with the message and uses SSL for confidentiality protection.

- *SAML Authorisation over SSL (SA)*: This profile attaches an authorisation token with the message and uses SSL for confidentiality protection. In this profile, the SAML token is expected to carry some authorisation information about end users. The sender of the token is actually vouching for the credentials in the SAML token.

- *SAML Sender Vouches with Certificates (SV)*: This profile protects messages with mutual certificates for integrity and confidentiality. The sender vouches a SAML token for authorisation.

- *SAML Holder of Key (HOK)*: Under this profile, the truest between a service and a client is not established directly, but requires the client to send a SAML assertion, issued by a specific SAML authority. The service has a trust relationship with the authority that issues the SAML token. The request is signed with the client's private key and encrypted with the server certificate. The response is signed using the server's private key and encrypted using the key provided within the SAML assertion.

- *Endorsing Certificate (End-Cert)*: This profile uses a symmetric key for integrity and confidentiality protection, and an endorsing client certificate to supplement the claims provided by the token associated with the message signature. For this profile, all the requests need to be authorised by a special identity, e.g. a purchase manager should endorse a purchase request to a vendor.

- *STS Issued Token*: This profile enables the use of a single Security Token Service (STS) to establish a chain of trust between servers and clients; especially where service providers and requesters are in different managed environments and confidentiality is a major issue. Instead of services trusting clients directly, services trust tokens issued by a trusted STS. The client then has to securely authenticate to this STS.

- *STS Issued Token with Service Certificate (STS-SC)*: This profile is similar to the previous one, except that confidentiality protection is enabled using a service certificate.

- *STS Issued Token Endorsing Token (STS-End)*: This is also an STS-based profile, that requires the client to authenticate using a SAML token that is issued by a designated STS, but the message signature has to be signed by an endorsing token.

2.11 The Performance of Web Services Security

The performance of the security mechanisms is fraught with concerns due to additional security contents in SOAP messages, the higher number of message exchanges to establish trust as well as extra processing time to process these additions. As the interaction between service providers and requesters occurs via XML-based SOAP messages, securing Web services tends to make these messages longer than they would be otherwise and consequently requiring interpretation by XML parsers on both sides, which reduces the performance of Web services (Menasce, 2002).

Performance is an open problem in Web services, and has been analysed in different manners. A study by Gray (2004) compared the performance of Web services with other middleware, such as CORBA and Java RMI. This study has shown that the performance of Web services is a major drawback. Another study (Jeckle, Melzer & Himsolt, 2004) also compared the same technologies and illustrated that the HTTP overhead causes higher response time for SOAP packages, which grows exponentially as the payload size increases.

The majority of the related studies have compared the performance of different toolkits. In (Machado & Ferraz, 2005), several SOAP toolkits have been evaluated with an objective of identifying and measuring SOAP inefficiency. Head et al. proposed a standard benchmark suite for quantifying and comparing the performance of the different SOAP implementations, such as *gSOAP*, *AxisJava*, *XSUL* and *bSOAP* (Head et al., 2005), and various XML parsers (Head et al., 2006).

Moreover, there have been several studies to benchmark the various aspects of performance by studying the effect of the implementation framework on the performance of Web services. For example, both studies by Sun Microsystems Inc. (2004) and Microsoft Corporation (2004) compared the performance of Web service technologies in two common middleware platforms: *Java 2 Platform Enterprise Edition (J2EE)* and *.NET*, that offer similar facilities for creating and using Web services. Similarly, Microsoft Corporation (2008) conducted a comparison of the performance of Web services applications deployed on two application servers (*.NET 3.5/Windows Server 2008* vs. *IBM WebSphere 6.1 ND/Red Hat Linux*). Each of the previous studies claimed that its framework has the upper hand in terms of Web services' performance; yet all the previous studies discussed the performance of plain-text services, without considering the security aspects of these services.

Early discussions of the security and performance trade-off (Vaughan-Nichols, 2002; Dodds, 2002) highlighted that SOAP-Web services suffer a performance hit when applying security measures. Because SOAP messages in their initial XML plain text are insecure, applying encryption and decryption on each message in the service-side and client-side will increase the overhead of these messages and increase processing time in both ends.

In security related studies, Shirasuna et al. (2004) evaluated three security approaches, namely *SSL*, *WS-Signature* and *WS-SecureConversation*, for grid services. Their evaluation has shown that transport level security is faster than message level security, and should be used if there is no additional requirement to use message level security. Their results indicated that *WS-SecureConversation* should be used when several messages are exchanged between the service and the clients, where XML-Signature is slightly faster than *WS-SecureConversation* for one-time invocations. Nevertheless, *WS-SecureConversation* has a scalability concern if the Web service is invoked by a huge number of clients simultaneously.

In a study of vertical scalability (i.e. adding capacity, such as *processors* and *memory*, to an existing system) of *Tomcat Application Server*, Guitart et al. (2005) examined the cost of security in Web services. However, its scope was restricted to the security of the communication channel, using *SSL*. Message layer security approaches were not considered in their tests.

Moralis et al. (2007) compared the performance of Web services with *Kerberos Token Profile* against *X.509 Token Profile*, while Liu, Pallickara & Fox (2005) conducted several tests for different operations (*Signing* vs. *Verifying* and *Encryption* vs. *Decryption* with several algorithms). Two studies by Tang et al. (2006) and Chen et al. (2007) compared the cost of *WSS Signing* and *WSS Encryption*. They indicated that using either the *Username* or *X.509* tokens does not make a significant difference to the performance of Web services.

Sosnoski (2009) studied how using the *WS-Security* and *WS-SecureConversation* standards affect the performance of Java Web services implemented using the *Apache Axis2* Web services stack. The aim was to provide a guideline on when and how to use *WS-Security*. He also suggested the usage of *WS-SecureConversation* for long-running exchanges of messages between clients and a server, especially when relatively small messages are exchanged between the service and its clients. Further work led by Sosnoski (2010) expanded that test to provide a performance comparison between the *Apache Axis2* and *Sun's Metro* Web services stacks. This experiment suggested that *Metro* is twice to three times faster than *Axis2* when security configurations (i.e. *signature*, *encryption* and *username tokens*) are applied, even though they perform similarly without security.

Aiming at providing a general guideline for selecting appropriate security mechanisms, the work by Novakouski et al. (2010) compared different *WS-Security* mechanisms (i.e. *Password Only, Sign Only, Encrypt Only. Sign Then Encrypt, Encrypt Then Sign,* and *WS-SecureConversation*) in details. It examined the impact of applying these mechanisms on the performance of SOAP-based Web services, measured in terms of: *Round Trip Time (RTT), message size* and *resource usage*. The study established that to minimise the huge penalty of applying security on the performance of Web services, a good understanding of the requirements and expectations is required, as there is no supreme standard that can provide security and performance in all applications and systems.

The previously discussed studies, however, considered the performance of the security standards that can be applied on Web services, such as the usage of encryption and digital signature, and the type security tokens. Alternatively, our study focuses on the overall performance of the security profiles, which simplify the security usage. Each profile predefines a set of security features to be implemented when securing a Web service. This approach shields the developers of Web services from the complexity of making a technical decision and allows them to focus on addressing the requirements of the security system.

2.12 Selecting Security Profiles

Architects and developers responsible for Web service security have a considerable number of options available to them. These options are further complicated by the fact that different projects and organisations have different security requirements. Applying different sets of standards and techniques could result in different quality measurements of a certain Web service (Shirasuna et al., 2004; Liu, Pallickara & Fox, 2005; Tang et al., 2006; Singhal, Winograd & Scarfone, 2007; Moralis et al., 2007;). To design, develop, and deploy secure Web services, architects and developers should be able to select an appropriate security profile amongst the available technologies by considering the new threats associated with exposing functionality on potentially unsecured networks (Microsoft Corporation, 2005), and simultaneously providing a level of quality that is acceptable by the service consumer (Casola et al., 2009).

Currently, there are several XML-based security profiles and mechanisms that may be used to satisfy the security requirements of a particular application. In an ideal situation, the task a Web service developer is to select the one profile that satisfies all the requirements. On many other occasions, there may be several solutions that satisfy most of the requirements, but not all of them. This can be considered as a decision problem, where an informed decision should be made by prioritising the requirements, and ranking the available options accordingly, to select the profile that matches the most important requirements.

The selection of a candidate solution depends on evaluating the available security profiles against several characteristics derived from the requirements of the given application. A Web services application developer specifies a set of requirements according to the service provider’s security preferences and technological constraints, taking into consideration the quality of service expected from the service consumer. The requirements can be divided into several dimensions according to different independent criteria and sub-criteria to create a multi-dimensional model. Each criterion and sub-criterion is given a weight according to its level of importance in the application. Each candidate solution is also weighted against each sub-criterion according to what extent this particular solution matches the requirement represented by this sub-criterion.

Generally, service providers set these weights on the basis of judgment. When there is a small number of selection criteria, then direct judgments may be a possibility. Nevertheless, when the number is large, the judging process may lead to improper selection due to the bias towards key elements only (Godse, Sonar & Mulik, 2008). In such a multi-criteria decision making model, it is important to have quantifiable values that are rational and consistent.

In research trends of Web services, the Analytical Hierarchy Process (AHP) is widely used as a Multi-Criteria Decision Making (MCDM) method to select the best Web service from a pool of services that have the same functionality provided by different service providers (Godse, Sonar & Mulik, 2008). Wu & Chang (2007) presented a conceptual model using AHP to help service consumers to find the service provider who provides the most optimal quality; yet their approach was not concerned with security.

The framework of Zuo, Wang & Wu (2008) approached the problem of selecting an optimal service among many Web services, which all meet the functional requirements, by establishing an index system for Web services products selection based on four aspects: *user, product, environment* and *supply*. They conducted a questionnaire survey to collect the views of 30 experts divided into two groups: business operation experts and academics.

Godse, Sonar & Mulik (2008) suggested the use of AHP as a quantitative approach to alternate the common ad-hoc practices in choosing Web services. Their model consists of three main criteria: *security*, *quality* and *business agreement*. Casola et al. (2009) also proposed a framework for quality and security evaluation. Their model depends on *response time*, *integrity* and *confidentiality* as measuring aspects to find a provider that guarantees these requirements. Thirumaran et al. (2011) proposed an AHP framework to choose the best custom search service based on *performance*, *cost*, *security* and *usability* requirements.

While all the previous models focused mainly on the point of view of the service consumer (selecting the optimal service among many available services that all meet the functional requirements), our research emphasises the service developer's viewpoint. The aim here is to provide a decision making tool to help Web services developers to select the best suited security approach to secure Web services that satisfies the security, configuration and performance requirements of a given application during the development process.

2.13 Summary and Discussion

This chapter provided background information in the area of Web services security and standards. The several approaches to analyse the performance of Web services security were discussed, which in turn led to highlight the high cost of applying security on the overall performance of Web services.

The literature review reports that the trade-off between security and performance depends largely on the selected security approach, where different security specifications affect the performance of Web services differently, as discussed by various studies (Shirasuna et al., 2004; Liu, Pallickara & Fox, 2005; Tang et al., 2006; Moralis et al., 2007; Chen et al., 2007; Sosnoski, 2009; Novakouski et al., 2010).

On the other hand, achieving a certain security level requires sometimes a combination of different security techniques (Hatala, Eap & Shah, 2004). There is no security standard that can achieve all the security requirements of all the applications while maintaining the best performance (Novakouski et al., 2010). As a result, a Web services developer would ideally apply a combination of security specifications in order to achieve the required level and coverage of security. This can be a very daunting task because of the complexity and variety of Web services security standards. Therefore, Several Web services development environments, such as *.net* and *JAVA's METRO* Web services stacks, provide developers with predefined sets of security specifications, or simply security profiles. The parameters of these profiles, for instance *security token*, *encryption algorithm* and *sign before/after encryption*, can be adjusted by the developers according to their security preferences.

In order to select the most suitable security profile for a given application, it is important to evaluate the security coverage of each profile against its performance. The previously mentioned related studies focused on the performance of individual security specifications and components. Instead, the approach of this research is to explore the overall performance of these profiles by developing a performance testing model to understand the cost of applying security profiles on the performance of Web services (Objective 1). The argument here is that understanding the overall performance of the security profiles will help in the initial selection process. Then, the performance of the individual security components (such as what proportion of that performance is due to the delay in encrypting the SOAP message and what proportion is because of acquiring credentials) can be considered to fine-tune the selected profile in order to achieve the finest solution.

The issue of selecting the best-fitted security profile can also be addressed by consulting rigorous documentations of best practices and case studies provided by reputable studies. However, due to the huge variation of the nature of modern systems that relay on Web services, different systems may have different requirements, configurations and limitations. Therefore, it is difficult to provide an ultimate solution that fits in all the scenarios. In addition, there are cases where more than one recommended solution can be implemented to satisfy the security requirements of a certain system, while there are other cases where there is no solution that satisfies all system requirements. Such cases require rating the various alternatives according to their coverage of the different requirements, and subsequently, selecting the security profile with the highest rate. This emphasises the need of a systematic framework to rate the alternatives according to different criteria (i.e. requirements) in order to aid Web services developers in selecting the most appropriate profile amongst a set of alternatives.

Several studies provided Multi-Criteria Decision Making (MCDM) frameworks for Web services discovery and composition (Wu & Chang, 2007; Zuo, Wang & Wu, 2008; Godse, Sonar & Mulik, 2008; Casola et al., 2009; Thirumaran et al.;

2011). Their approaches considered the service consumer point of view (the aim is to select the most suitable Web service from a pool of services that provide the same functionality). Alternatively, we approach the issue from the Web services developer view-point by addressing how to provide a Web service with the best-possible security that considers the performance expectations and the service provider's limitations (Objective 2).

The MCDM frameworks introduced in previously discussed related work were illustrated via the use of example scenarios. However, they did not provide an empirical validation of their efficiency as they did not provide a benchmark to validate against. In this thesis, we try to validate and evaluate the developed framework using a scenario-driven approach. To demonstrate the framework's utility and value, the tested scenarios are chosen from well-known documentations of best practices, and the results of the framework are compared to the recommendations of these documents (Objective 3).

Chapter 3: Research Design and Approach

3.1 Overview

This chapter aims at describing the Design Science Research (DSR) methodology which will be undertaken as a general research methodology for this thesis. A detailed discussion from its different perspectives is presented in order to explain and justify the adoption of this methodology.

3.2 Philosophical Grounding

A paradigm can be defined as a set of basic beliefs which guide the actions and the activities of the researchers throughout the research process (Guba & Lincoln, 1994; Mingers, 2001). Research in the field of information systems and computing can be categorised into three main paradigms (Chua, 1986; Orlikowski & Baroudi, 1991; Klein & Myers, 1999; Vaishnavi & Kuechler, 2009). These paradigms are:

- The Positive Research: Collected data is used to support hypotheses and assumptions that made prior to investigation.
- The Interpretive Research: Collected data is used to extract knowledge without making assumptions.
- The Design Science Research approach: Constructing artefacts and evaluating them is used to explain and enhance aspects of the system.

In the research world, there are four philosophical theories to view the research paradigms (Guba & Lincoln, 1994; Mingers, 2001; Vaishnavi & Kuechler, 2009):

- The theory of existence (Ontology) which describes the nature of reality by asking questions like what is real and what is not? What is fundamental and what is derivative?
- The theory of knowledge (Epistemology) that explores the nature of valid or true knowledge.
- The theory of reasoning and inference (Methodology) which studies the relations between theory and practice in order to identify the best approach that helps in generating the desired knowledge in a valid and reliable way.
- The theory of value and value judgement (Axiology or Ethics): What is of value or considered right?

Table 3-1 summarises the theoretical perspectives of these four research paradigms.

Our research aims at producing an artefact in the form of a framework to help architects and developers to choose the best-suited security profiles for Web services applications. The aim of this research is highly consistent with the general aim of DSR. This is because the research in information systems and computing is considered a DSR research, if the main aim is to change a current situation related to organisational or social systems into a more desirable one through the development of novel artefacts (Hevner et al., 2004). Hence, we argue that DSR is highly fitting in the context of this research.

Indeed, design-science research is primarily a *problem solving* paradigm (Hevner et al., 2004) that seeks to *create* artefacts addressing the so-called *wicked problems* (Pries-Heje & Baskerville, 2008; March & Storey, 2008). In principle, design-science research attempts to successfully design, develop, and evaluate technology-oriented design artefacts characterised as novel, innovative, and purposeful. When portrayed as *purposeful*, this implies that these artefacts would potentially provide organisations and humans with recognisable utility since they should address unsolved problems (Hevner et al., 2004), or provide better solutions and thus enhance existing practices (Kuechler & Vaishnavi, 2008). Hence, these artefacts aim to provide additional improvements to real-world phenomena (March & Storey, 2008; Iivari, 2007; Purao, 2002). Therefore, while humans could change their life styles through the introduction of these novel artefacts, organisations might change the ways in which they do business so as to exploit the opportunities that emerged due to these artefacts.

	Research Perspective		
Basic Belief	**Positivist**	**Interpretive**	**Design**
Ontology	A single reality. Knowable, probabilistic	Multiple realities, socially constructed	Multiple, contextually situated alternative world-states. Socio-technologically enabled
Epistemology	Objective; dispassionate. Detached observer of truth	Subjective; values and knowledge emerge from the researcher-participant interaction	Knowledge through making: objectively constrained construction within a context. Iterative circumscription reveals meaning
Methodology	Observation; quantitative, statistical	Participation; qualitative. Hermeneutical, dialectical	Developmental: Measure artefactual impacts on the composite system
Axiology	Truth: universal and beautiful; prediction	Understanding: situated and description	Control; creation; progress; understanding

Table 3-1: Philosophical Assumptions of Research Approaches

(Vaishnavi & Kuechler, 2009)

3.3 Overview of Design Science Research

Design Science research (DSR) is a set of analytical techniques and perspectives for performing research in the area of information systems and computing. Design Science Research involves the analysis of the use and performance of designed artefacts to understand, explain and most probably enhance the behaviour of aspects of information systems (Vaishnavi & Kuechler, 2009).

Design means to invent, plan and develop something for particular purpose. To bring design activity into focus at an intellectual level, we should distinguish between “natural science” and “science of artificial”. The natural science is the knowledge about objects or phenomena that describes and explains how they behave and interact with each other. On the other hand, the knowledge about artificial objects and phenomena designed to meet certain desired goals is known as the science of artificial.

Research, according to Kuhn (1996), is defined as an activity that contributes to the understanding of a phenomenon. A phenomenon is a set of behaviours of some entities that is found interesting by the researcher. In the case of information systems, the phenomenon, or part of it, may be created, instead of naturally occurring. Understanding means the knowledge that allows prediction of the behaviour of some aspects of the phenomenon. Research methods or techniques are the set of activities a research community considers appropriate to the production of understanding.

Owen (1997) introduced a general model for generating and accumulating knowledge (Figure 3-1) that is helpful to understand the Design Science Research process: “Knowledge is generated and accumulated through action. Doing something and judging the results is the general model” (Owen, 1997). The process is illustrated as a cycle where knowledge is used to create works, and works are appraised to create knowledge.

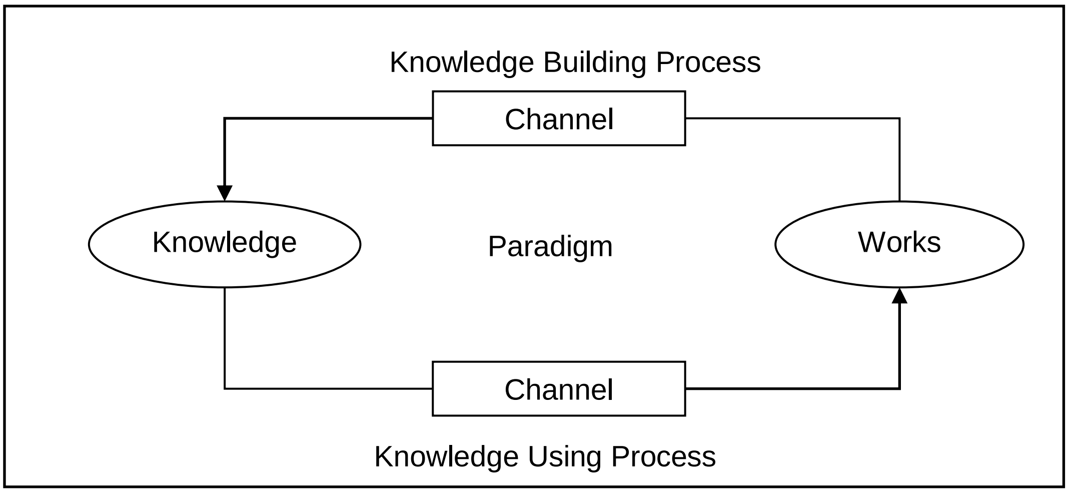

Figure 3-1: A General Model for Generating and Accumulating Knowledge

(Owen, 1997)

3.4 The Outputs of Design Science Research

Based on the work of March & Smith (1995) and Hevner et al. (2004), Vaishnavi & Kuechler (2009) have proposed five general outputs explicate the types and levels of knowledge that can be derived from Design Science Research, highlighted in Table 3-2.

Output	Description
Constructs	The conceptual vocabulary of a domain
Models	A set of propositions or statements expressing relationships between constructs.
Methods	A set of steps used to perform a task (how-to knowledge).
Instantiations	The operationalisation of constructs, models and methods.
Better theories	Artefact construction as analogous to experimental natural science

Table 3-2: The Outputs of Design Science Research

(Vaishnavi and Kuechler, 2005)

Purao, (2002) has pointed out a different perception on the output of Design Science Research where the multiple outputs are classified by level of abstraction, as shown in (Figure 3-2).

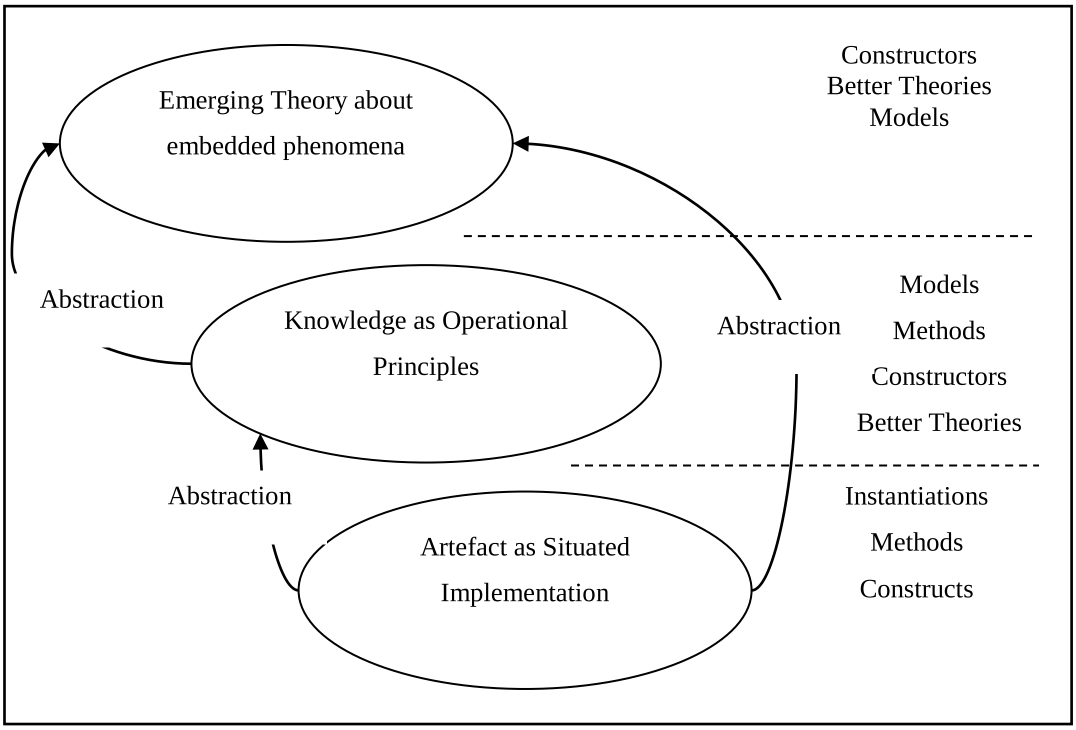

Figure 3-2: Outputs of Design Science Research (Purao, 2002)

3.5 Design Science Research Methodology

Drawing heavily from the work of Owen (1997), Vaishnavi & Kuechler (2009) have developed a general methodology of Design Science Research, (Figure 3-3). In this model, each of the stages can be revisited at any point in the process. Therefore, Design Science Research in considered as an interactive approach, which is especially suitable for software development, because requirements are constantly changing, and findings from one stage may require a revisit to a previous stage to alter the design of the system and improve it.

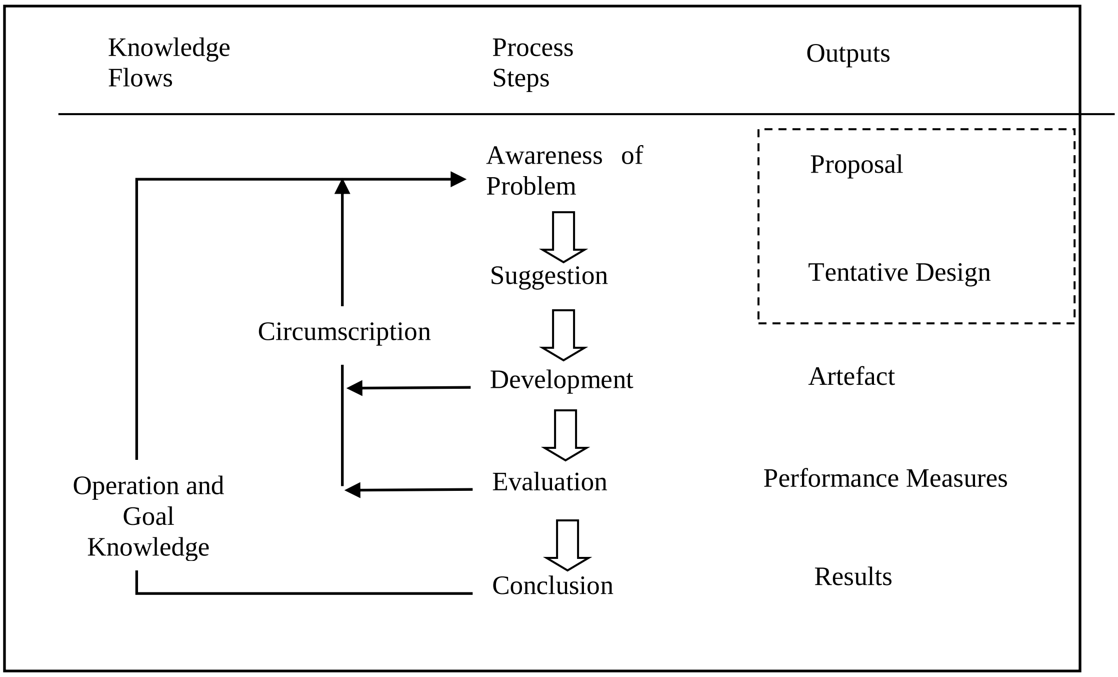

Figure 3-3: The General Methodology of DSR (Vaishnavi & Kuechler, 2009)

The previous figure illustrates the five stages of the Design Science Research cycle, where "knowing is making", and each stage can be revisited at any point in the process. The following section will discuss these stages (Vaishnavi & Kuechler, 2009):

Awareness of Problem: This may come from multiple sources: new developments in industry or in a reference discipline. Reading in an allied discipline may also provide the opportunity for application of new findings to the researcher's field. The output of this phase is a *proposal*, formal or informal, for a new research effort.

Suggestion: This stage comes immediately after the *proposal*. The output of this phase is a *tentative design* that is intimately connected with the *proposal*.

Development: The provisional Design is implemented in this stage and the final result is an *artefact*.

Evaluation: Once constructed, the *artefact* is evaluated according to criteria that are always implicit and frequently made explicit in the *proposal* (Awareness of Problem phase).

Conclusion: This phase is the final effort of a specific research. Normally, if the result is satisfying, even if there are still deviations in the behaviour of the artefact from the hypothetical predictions, the results may be considered *good enough*. The results of the effort are consolidated and written up at this phase, and the knowledge gained in the effort is frequently categorised as *firm facts* that have been learned and can be constantly applied, or behaviour that can be repeatedly invoked. Otherwise, the knowledge gained in the effort is categorised as *loose ends* or uncharacteristic behaviour that requires explanation and may well serve as the subject of further research.

3.6 Developing a Framework for Selecting Security Approaches Based on DSR

Following the DSR paradigm, we aim to develop a framework for selecting a Web services security approach that is most appropriate for a certain application. To this end, three possible approaches are investigated throughout the DSR cycle, namely AHP, performance analysis and steganography.

By referring to (Saaty, 2008), we recognise four facets need to be defined to develop the first approach: (a) goal; (b) alternatives; (c) criteria; and (d) sub criteria. An AHP hierarchy indicates a relationship between elements of one level with those of the level immediately below. This relationship percolates down to the lowest levels of the hierarchy. In the hierarchic structure, at the root of the hierarchy is the goal or objective of the problem being studied and analysed. The leaf nodes are the alternatives to be compared. In between these two levels are various criteria and sub-criteria. It is important to note that when comparing elements at each level, a decision-maker needs just to compare with respect to the contribution of the lower-level elements to the upper-level ones.

In the context of this research, defining the goal of the AHP framework was quite straightforward as it is directly mapped to the current research problem; i.e. choosing the best-suited security profile to be deployed for a particular application including its Web services. Having the goal established, we moved a step further to define the decision alternatives. For this purpose, we selected the security profiles that are deemed representatives.

However, the issue of defining the decision criteria and sub-criteria was more complex. Three iterations incorporating *design*, *deployment*, and *evaluation* courses of action were needed before the final artefact (i.e. AHP Framework) has been emerged which includes a comprehensive criteria and sub-criteria in this context. In addition, a fourth iteration has been used to introduce an alternative approach to secure Web services based on steganography. In the following sub-sections, we discuss these four iterations in detail.

3.6.1 DSR Iteration One: Library Research

In this iteration, we followed a library research in which a comprehensive literature review was undertaken. The main purpose of this library research was to understand the security elements or criteria that are relevant in the domain of Web services. In fact, this iteration was not very challenging given the fact that there is almost a consensus in the relevant literature regarding Web service security measures (Nakamur, Hada & Neyama, 2002; Yang, 2002; Geer, 2003; O'Neill et al., 2003; Siddiqui, 2003; Naedele, 2003; Rao et al., 2004; Tatsubori, Imamura & Nakamura, 2004; Nadalin et al., 2006/2007). The identified security criteria based on which Web service security profiles need to be compared are: (1) *Authentication*, (2) *Integrity*, (3) *Confidentiality*, (4) *Non-repudiation*, (5) *Authorisation,* and (6) *End-to-End Security*.

After establishing these criteria, we deployed and utilised them to compare different Web service security profiles. Retrospectively, we found out that although these criteria facilitate reducing the number of favoured security profiles for a certain application; yet there is a number of security profiles that can be employed, such as: *UA*, *UDP*, *MCS*, *SSL*, *SA*, *SV* and *STS*. Consider this example, a certain Web application is highly critical and requires fulfilment of all of the previously discussed security criteria. To choose the best-suited security profile, the developer need to compare the security profiles based on the defined security criteria. These criteria alone will lead the developer to find out that more than one security profile can be selected, although there are slight differences in the level of security they can achieve.

Moreover, despite the fact that these security profiles can achieve the defined security requirements to a certain degree, they may have substantial differences in terms of performance. Hence, we recognised that these criteria alone are not sufficient for developers so as to take informed decisions and that performance requirements need to be taken into consideration. Table 3-3 summarises this iteration. This evaluation led us to start the iteration two of our DSR research.

Problem:	Selecting security approaches for Web services.
Suggestion:	To identify and define: 1. The security criteria and sub-criteria. 2. The available security profiles.
Outputs:	***Model:*** Initial selection framework (based on a decision making approach). ***Construct:*** security criteria (e.g. authentication, authorisation) and security profiles (e.g. UA, MCS). ***Method:*** Initial AHP framework (security criteria).
Evaluation:	The performance evaluation of the security profiles is required.

Table 3-3: DSR Iteration One (Library Research)

3.6.2 DSR Iteration Two: Laboratory Experiments

To test the effect of individual security profiles on Web services performance, we designed and implemented a performance testing experiments, where a simple Java API for XML Web Services (JAX-WS) application was used. This Web application consists of a Web service and a client, and it represents the peer-to-peer mode test. The performance analysis framework resulted from this iteration represents the first approach to consider when selecting a security profile since it provides performance guidelines of the security profiles.

Problem:	Selecting security approaches for Web services.
Suggestion:	Analyse the performance of the identified security profiles.
Outputs:	***Model:*** Improved selection framework (based on performance analysis and decision making approaches). ***Construct:*** performance criteria (e.g. Round Trip Time, message size). ***Instantiation:*** Performance tests. (Chapter 4:). ***Method:*** AHP framework (security and performance criteria).
Evaluation:	The configuration limitations and preferences of the system should be considered.

Table 3-4: DSR Iteration Two (Laboratory Experiments)

Having reached this point of the research, we recognised that security profiles need to be selected not only based on their fulfilment of security requirements, but also based on their performance measures. In other words, the AHP hierarchy at this point included two major criteria (security and performance) along with their sub-criteria. By utilising the current AHP hierarchy we started analysing different scenarios and cases related to Web applications. This step has highlighted an important criterion which has been overlooked, namely configuration requirements. In fact, while conducting these cases, it became apparent that different security profiles normally call for different infrastructures to be in place.

Examples of these infrastructure and configuration requirements include certificates and security service tokens. Table 3-4 illustrates the second iteration.

Having recognised the importance of configuration requirements to the decision of which security profile is most appropriate to be deployed in a certain situation; we started DSR iteration 3 to further enhance the AHP framework.

3.6.3 DSR Iteration Three: Configuration Requirements

As highlighted in the previous subsection, each security profile requires configuring some options on the Web service host. These configuration requirements reflect the technological constrains and system preference of the Web service provider. The Web service client may need to be configured depending upon the security profile selected by the server side. For Web service security profiles, the service configuration requirements are: (1) *Certificates stores*, (2) *Security Token Service*, (3) *Users' database*, and (4) *Flexibility*.

After these three iterations of the conducted design-science research, we have reached a point where, from our perspective, the developed AHP framework in its current form can be implemented and tested using real scenarios to verify its utility and value. The results of the evaluation stage indicate that the AHP framework can be used to facilitate the selection of a security profile based on the three selected criteria: *security*, *performance* and *configuration*. Table 3-5 explains the third DSR iteration.

On the other hand, the performance results, from iteration two, highlighted the effect of applying security and reliability profiles on the size of the secured SOAP messages and related that to the drop of performance of Web services. Therefore, in iteration four, we look at another alternative to secure Web services using steganography rather than encryption in order to minimise the impact of the message size.

Problem:	Selecting security approaches for Web services.
Suggestion:	Analyse the configuration requirements of the identified security profiles.
Outputs:	***Model:*** Improved selection framework (based on (1) performance analysis and (2) improved decision making approaches). ***Construct:*** configuration criteria (e.g. security tokens, users' database). ***Method:*** Final AHP framework (security, performance and configuration criteria). (Chapter 5:) ***Instantiation:*** AHP tool. (Appendix C)
Evaluation:	- The AHP framework is evaluated using scenarios from documentations of best practices. The results are satisfactory. - Alternative approaches may be considered to reduce the size of security assertions.

Table 3-5: DSR Iteration Three (Configuration Requirements)

3.6.4 DSR Iteration Four: Alternative Solution Based on Steganography

The literature review, as well as the performance analysis conducted as part of this research, demonstrates that applying security and reliability profiles to SOAP messages decreases the performance of Web services dramatically. This is due to the extra security assertions added to these messages and the higher number of message exchanges needed to establish trust and reliability. Therefore, we have considered alternative options, using steganography, to establish trust between Web services and their clients. During this iteration, we developed a method to embed a hidden message in the sender endpoint and extract this message in the receiver endpoint. This method is based on shuffling the XML tags of a SOAP message in a particular order, where each permutation of these tags has a specific meaning (e.g. an alphabet letter or a control status).

Problem:	Selecting security approaches for Web services.
Suggestion:	Explore and evaluate the feasibility of using steganography as an alternative approach to secure Web services.
Outputs:	***Model:*** Final selection framework (based on (1)performance analysis, (2)decision making and (3)steganography approaches). ***Method:*** Embedding and Extracting Algorithms. (Chapter 6:)
Evaluation:	The steganography method is validated using an example to illustrate its utility and value. The results are satisfactory. The overall selection framework is accepted.

Table 3-6: DSR Iteration Four (Alternative Solution Based on Steganography)

After the fourth iteration (Table 3-6), the final selection framework is considered to be satisfactory. This framework provides three approaches to tackle the problem of selecting a suitable security approach for a given Web services application. The first is to use the results of the performance analysis instantiations that have been developed during iteration 2. Those instantiations map the performance aspects to the tested security profiles and can be used when there is no need to consider the system limitations and constraints. The second approach is to use the AHP multi-criteria decision making framework to make an informed decision based on the security, performance and configuration criteria. This approach enables developers to rate the tested security profiles according to those criteria. Finally, the third approach is to apply our developed steganography method. This method is an alternative to traditional encryption-based security approaches and can be used to minimise the size and the number of exchanged messages.

3.7 Summary

This research is about constructing an artefact (A multi-criteria decision making framework) and evaluating it as an effort to enhance and automate the process of selecting the most appropriate Web services profile in a given application. Accordingly, we are dealing with an artificial science (information systems and computing) rather than a natural science. Therefore, Design Science Research is the obvious choice as a general methodology for this research. This research aims, by applying the DSR, at producing an artefact in the form of a framework aiming to help Web services developers to select the best-suited security profiles for software applications. The aim of this research is highly fitting with the general aim of DSR. The research in information systems and computing is considered a DSR research, if the main aim is to change a current situation related to organisational or social systems into a more desirable one through the development of novel artefacts (Hevner et al., 2004). Hence, the DSR is highly consistent in the context of this research.

Chapter 4: A Performance Evaluation of Security Profiles for Web services

4.1 Overview

This chapter describes a series of performance tests that focuses on understanding the impact of applying various security profiles on the performance of Web services. The collected results represent a starting point for understanding trade-offs between performance and security, and form a basis for making architectural and engineering decisions, which will be discussed in the following chapter.

Chapter 4 is organised as follows: The test design is described in section 4.2. The test results are presented and analysed in section 4.3. Finally, section 4.5 summarises the key findings.

4.2 Test Design

Recently, Web services security has witnessed a significant impetus as several specifications have been developed and implemented to meet the security challenges of Web services. However, the performance of the security mechanisms is fraught with concerns due to additional security contents in SOAP messages, the higher number of message exchanges needed to establish trust as well as extra CPU time to process these additions. *See Appendix A for a comparison between four SOAP messages (Simple, Secured with UA, Secured with MCS, and Reliable Message) with an initial data of one character.*

This test focuses mainly on the overall performance of WSIT security and reliability profiles. Therefore, the following discussion does not essentially cover the performance of underlying WSS specifications implemented within the security profiles.

4.2.1 Test Subjects

This section describes the METRO/WSIT project (Sun Microsystems Inc., 2007) and its security profiles that were selected to be benchmarked by the experiments.

In order to improve Web services' Quality of Service (QoS) and to enable interoperability between Java and .Net Web services, the Web Services Interoperability Technologies (WSIT) (Sun Microsystems Inc., 2007) has been developed as joint effort between Sun and Microsoft. WSIT is an implementation of a number of open Web services specifications to support enterprise features, such as message optimisation, reliable messaging, and security.

Web services have relied on transport-based security such as SSL to provide point-to-point security. WSIT implements WS-Security so as to provide interoperable message content integrity and confidentiality, even when messages pass through intermediary nodes before reaching their destination endpoint. WS-Security, as provided by WSIT, is an addition to the existing transport-level security, which may still be used.

WSIT provides a number of security profiles that can be applied to secure Web services. Each profile represents a set of pre-defined security specifications and configurations. Using security profiles reduces the development time and allows Web services developers to focus their effort on identifying the security requirements of their systems rather than going into the complexity of understanding the several security standards and finding the right combination that fulfils the security needs of the developed systems. However, while applying WSIT security profiles to enhance the security of Web services, this may also result in increasing the size and number of the exchanged SOAP messages, which may in turn lead to an increase in the time of processing these messages and transmitting them over the network.

Currently, there are many security profiles that can be implemented to secure Web services, see section 2.10.3 . We have deliberately selected seven of them for this experiment. The selection was based on identifying the generic profiles, which cover the general configuration types and represent the different security methods, as illustrated in Figure 4-1. The selected profiles are:

- Username Authentication with Symmetric Key (UA).
- Username with digest passwords (UDP).
- Mutual Certificates Security (MCS).
- Transport Security using SSL (SSL).
- SAML Authorisation over SSL (SA).
- SAML Sender Vouches with Certificates (SV).
- STS Issued Token (STS).

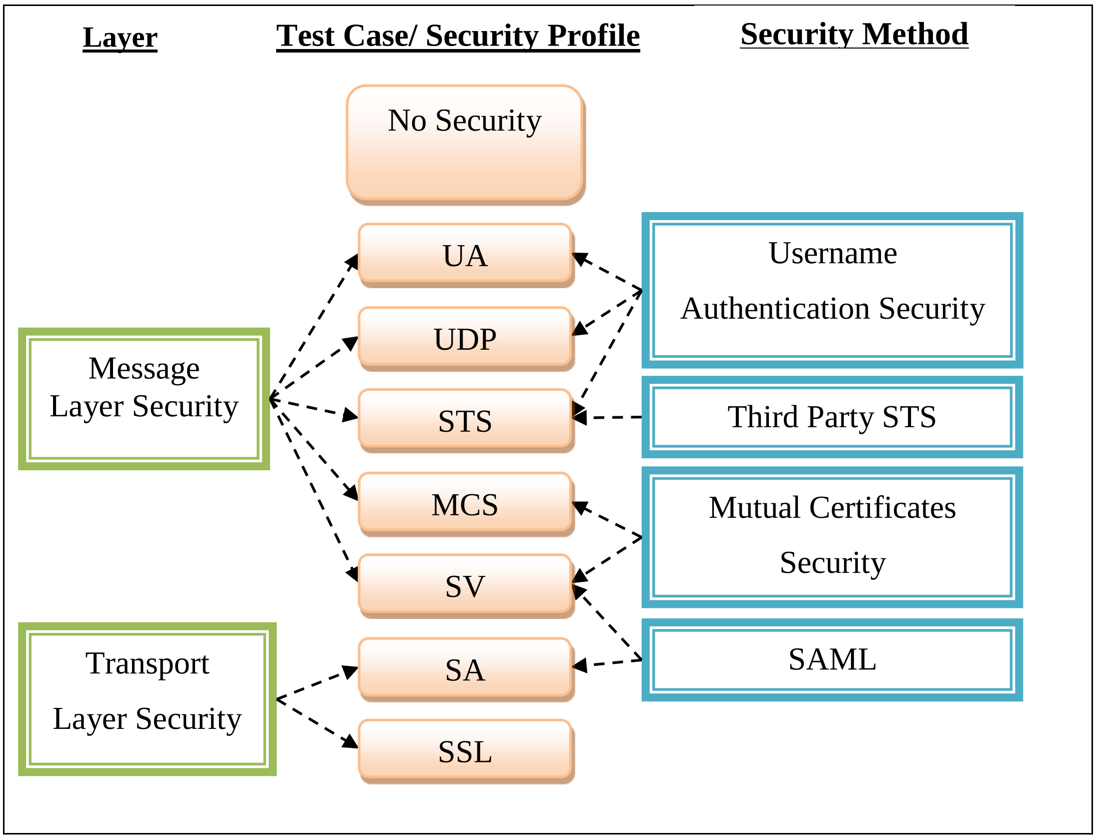

Figure 4-1: METRO Security Profiles

We also test the impact of applying *Reliable Message Delivery* and *Deliver Messages in Exact Order,* as provided by Metro's WSIT, on the performance of Web services.

4.2.2 Test Scenario and Cases

Echo Web service (Simple structure/ Dynamic payload): This JAX-WS echo application consists of a Web service and a client, which represents the peer-to-peer mode test; the client sends different size auto-generated messages (from 1 Byte to 1MByte) and the Web service echoes (send back) the same message received. The test was run with and without applying security profiles, using different initial message sizes: 1byte to 1 Mbyte. This Web service was used to test the performance of security profiles because using a simple payload reduces the side effects of unrelated processing of the business logic. In addition, as security profiles employ encryption algorithms, which are used to decipher the exchanged data, the encryption/decryption process depends on the size of the message. Therefore, a dynamic size payload was selected for this experiment.

Book details Web service (Complex structure/ static payload): The client sends a sequence of messages; each contains a one-element array of book details objects (Figure 4-2). The Web service replies by sending a simple string response for each message it receives. The complex structure/ static payload was used for the performance of reliable messaging methods because reliability guarantees the delivery of the message, as a whole, regardless of the actual payload. If a message is lost, the sender resends the message until its receipt is acknowledged by the receiver. If these messages are received out of order, the receiver can rearrange the messages into the correct one.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns2:bookOrder xmlns:ns2="http://service.testproject/">
   <book>
     <isbn>1-11-111111-1</isbn>
     <author>Author_1</author>
     <name>Book_1</name>
     <pages>111</pages>
     <publisher>Publisher_1</publisher>
   </book>
  </ns2:bookOrder>
 </S:Body>
</S:Envelope>
```

Figure 4-2: Complex Payload SOAP-Message

4.2.3 Test Environment and Settings

In this chapter, the focus is on the increment of processing time when applying security profiles instead of the network latency. As a result, the data were collected from a local machine; the Web service and the client were deployed on a Dell machine (Pentium D CPU 2.80 GHz / 3GB of RAM) Running Microsoft XP.

NetBeans IDE 6.5 was used to develop the Web service and the client. The Web service was developed as a Web application and deployed on a GlassFish 2.2 application server. A Java SE application was used to represent the client. The initial data sent from the client to the service were randomly generated before sending the message to avoid any caching. Metro’s WSIT Web service stack 1.4 was used to apply security profiles to the tested Web service.

4.2.4 Evaluation Metric

The time spent in requesting and responding on the client side was measured as round trip time (RTT) using *Java’s System.nanoTime()*. For the reliable messaging experiment, we also measured the average response time, maximum response time and the maximum throughput using the Web services monitor in the *Glassfish Admin Console*. Every test was repeated 1000 times and the average, maximum and standard deviation RTT were calculated for each case. The test is then repeated on 10 different occasions, and used the average results after eliminating the highest and lowest scores to reduce the noise results. The results were then compared using the Round Trip Time Increment Percentage (Tang et al., 2006; Chen et al., 2007) (RTTIP) in order to evaluate the performance overhead for a specific security profile deployment:

$$RTTIP = \frac{RTT_i - RTT_0}{RTT_0} \times 100\% \qquad (1)$$

Where:

- RTT_0 is the round trip time without applying any security profile deployment.
- RTT_i is the round trip time of the Web service with a specific security profile i deployment.

4.3 Results and Analysis

The average value of the RTT for each initial message size and each profile is used to study the normal behaviour of the Web service (Table 4-1). We also calculated the standard deviation RTT (Table 4-2) and maximum RTT (Table 4-3) as suggested by (Casola et al., 2009) to indicate how often the Web service secured using a certain profile shifts from its normal behaviour, and the worse scenario, respectively.

Security Profile	Initial Message Size (Byte)						
	1	10	100	1 K	10 K	100 K	1 M
No Security	24	25	25	25	27	42	228
UA	80	81	82	84	101	263	2055
UDP	80	80	80	81	98	259	2051
MCS	119	119	119	121	137	295	2109
SSL	42	42	42	43	45	71	348
SA	52	52	53	52	55	80	357
SV	122	122	124	124	140	303	2110
STS	266	285	288	294	316	481	2229

Table 4-1: Average Round Trip Time (milliseconds)

Security Profile	Initial Message Size (Byte)						
	1	10	100	1 K	10 K	100 K	1 M
No Security	6	8	8	9	12	31	54
UA	25	28	28	28	33	59	18
UDP	26	26	27	27	31	57	15
MCS	29	29	29	29	33	56	16
SSL	22	22	22	23	24	42	20
SA	23	24	23	23	25	41	24
SV	23	24	27	24	29	57	15
STS	664	957	1048	1246	1272	1201	498

Table 4-2: Standard Deviation Round Trip Time (milliseconds)

Security Profile	Initial Message Size (Byte)						
	1	10	100	1 K	10 K	100 K	1 M
No Security	207	213	211	211	211	211	390
UA	237	237	236	239	253	400	2509
UDP	235	235	235	237	251	393	2398
MCS	295	274	275	277	291	435	2489
SSL	205	205	205	206	205	218	381
SA	215	214	215	215	214	226	460
SV	282	282	282	282	298	516	2464
STS	19218	23250	22973	30351	30366	26838	33229

Table 4-3: Maximum Round Trip Time (milliseconds)

The data collected from these experiments can be analysed in many different ways. Figure 4-3 illustrates the RTT Increment percentage for the average values of the tested security profiles. This figure shows three types of performance behaviour in response to the increase in the message size. The first group includes SSL and SA, which are transport layer security profiles. These profiles are clear winners as they have the smallest incremental percentage values amongst the rest, and their RTTIP decreases when the data size increases. However, they provide point-to-point security only and cannot guarantee end-to-end security. The second group includes UA, UDP, MCS and SV. These message layer security profiles can provide end-to-end security, but they show a significant increase in their RTT as the data size increases. The third group is for the STS-based profiles, which have the worst performance amongst all the other profiles. However, their performance is stable for all data sizes, and becomes very similar to the rest of the message layer security profiles for large data sizes (1 MByte+).

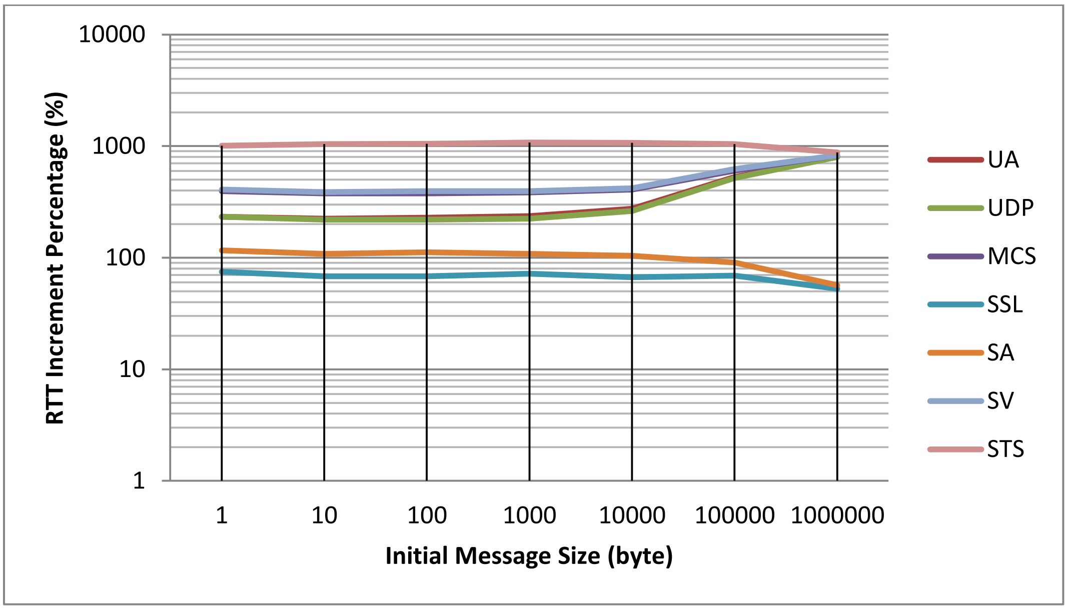

Figure 4-3: RTT Increment Percentage for the Average Values

Based on the previous observations, the results can be discussed using the following criteria: security layer (transport vs. message), encryption type (symmetric vs. asymmetric), the usage of SAML tokens and finally authentication type (direct vs. STS).

4.3.1 Transport Security vs. Message Security

Figure 4-4 illustrates the huge difference in performance between message level security, represented by UA, and transport level security using SSL. While the increment percentage of RTT using message level security increases when the data size increases, we can notice its decrease when using transport layer security. This is because SSL is lightweight, as it depends on securing the communication channel and does not involve any XML parsing. Therefore, transport layer security should be used if there is no special requirement to use message level security, such as having a chain of Web services, where end-to-end security is required.

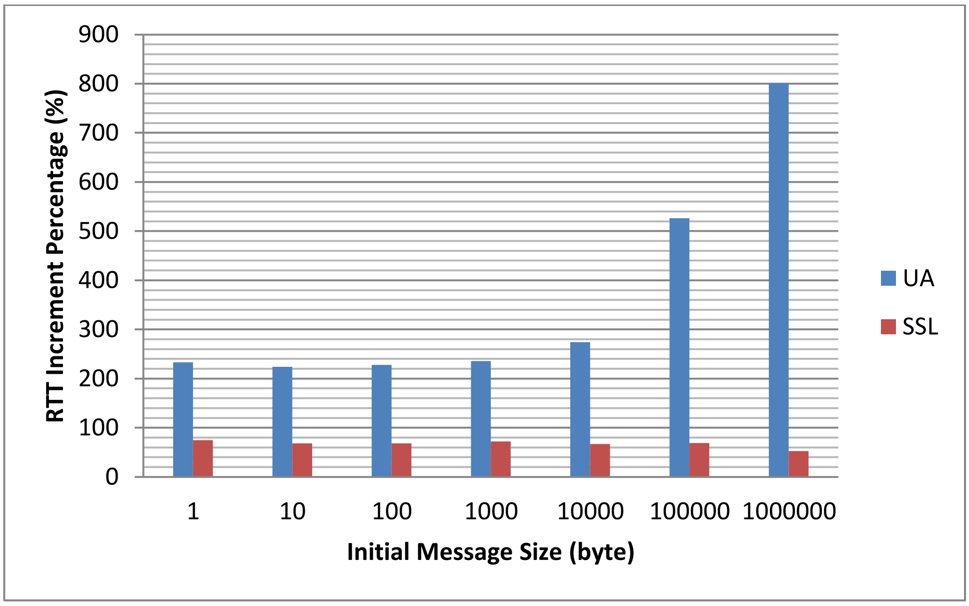

Figure 4-4: Transport Security vs. Message Security

4.3.2 Username Tokens vs. Mutual Certificates

As shown in Figure 4-5, when the initial data size is in the range of 1 byte to 1 Kbyte, the round trip time is increased by around 220-230% when applying username authentication profiles. The UDP performs slightly better than UA because the username token in UDP, unlike UA, is not encrypted due to the use of digest passwords. On the other hand, using MCS for the same data sizes increases the round trip time by 370-390%. The difference can be related to the fact that UA and UDP use symmetric key cryptography while MCS uses asymmetric cryptography, where symmetric is always faster than asymmetric encryption (Sun Microsystems Inc., 2007).

When large messages are exchanged between the client and the service (i.e. 1Mbyte) we notice however that the difference between the performance of UA, UDP and MCS decreases dramatically because most of the processing time is spent on applying the actual encryption rather than manipulating the keys.

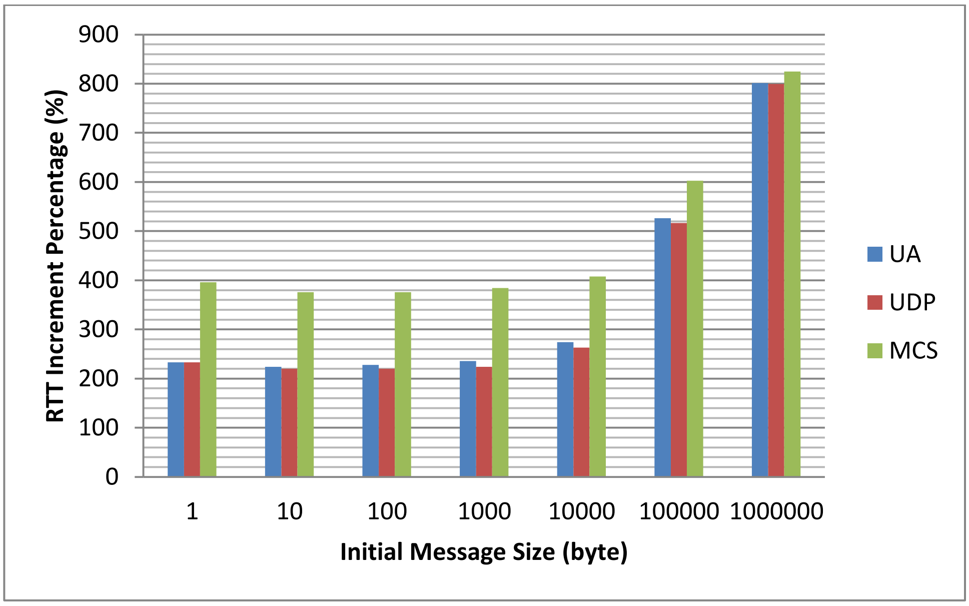

Figure 4-5: Username vs. Mutual Certificates

4.3.3 SAML: Over SSL vs. Mutual Certificates

Figure 4-6 confirms that the performance of SAML-based security profiles (SA and SV) depends mainly on the underlying security method that is used to protect the data (SSL and MCS respectively).

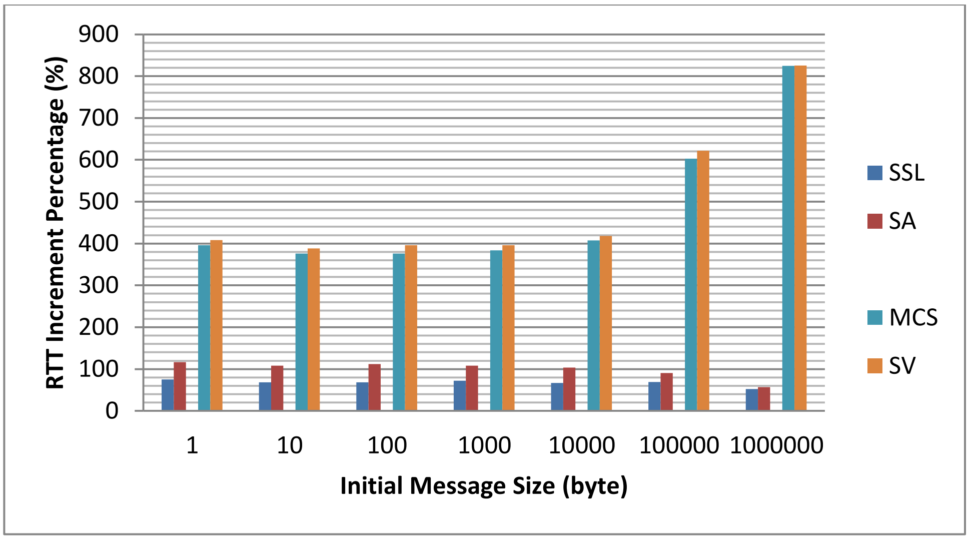

Figure 4-6: SAML-Based Profiles Compared to Their Underlying Security Profile

In both of the security profiles, SA and SV, the sender vouches a SAML token for authorisation. However, SA protects the exchanged message using SSL in the transport layer, while SV depends on mutual certificates, which is a message level security mechanism. Comparing the performances of (SSL vs. SA) and (MCS vs. SV) indicates that there is a negligible increase in RTTIP when SAML is applied, especially when very large messages are exchanged.

4.3.4 STS vs. Non-STS

In Figure 4-7, we compare the performances of STS-based and non-STS (direct client-service authentication) security profiles. The non-STS based profile used for the comparison is UA. Since the security of the client is dependent upon the security profile selected for the STS itself, not the service, the STS itself is secured using a separate UA profile.

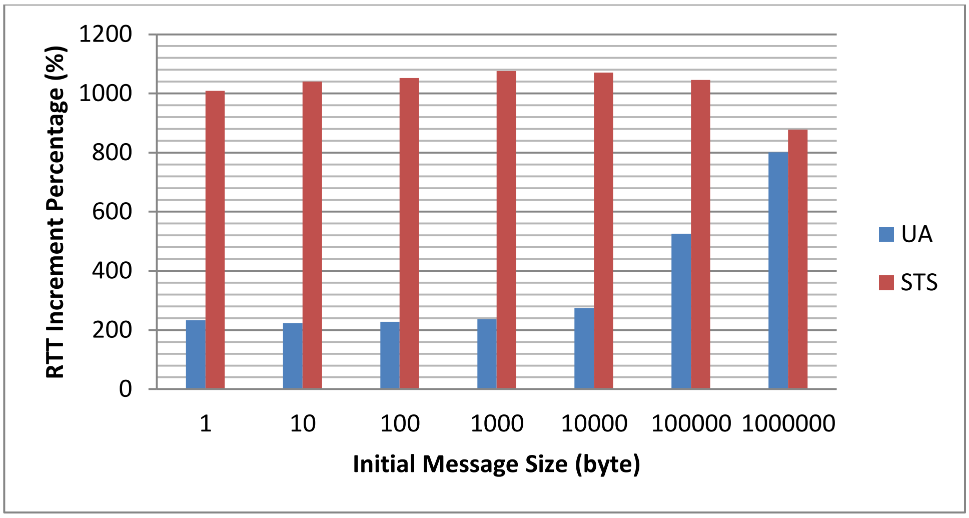

Figure 4-7: STS vs. Non-STS

Results show that using third party STS tokens affect the performance of the Web service significantly. When the initial message size is between 1 byte – 1 Kbyte, the RTTIP of STS is about 4 times its value when applying non-STS security profile, UA. Therefore, STS security profiles should only be used when the service and the client are in two different domains, where direct authentication can be an issue. However, when the initial data size reaches 1 M byte, the performance difference decreases noticeably.

4.3.5 Average, Standard deviation and maximum

This section explores the effect of changing the data size on the performance behaviour of the security profiles.

In order to demonstrate the normal behaviour, the average values of RTT were used, as shown in Figure 4-8. All the testes profiles show a very small increase in the RTT when the data size increases gradually from 1 byte to 10 Kbyte. However, the increase becomes significant when the data size is increased to 100 Kbyte and then 1 Mbyte. The message layer security-based profiles (UA, UDP, MCS and STS) respond with a very sharp increase in the RTT when the data size reaches 1 MByte. Nonetheless, the differences between the RTT values of these message layer security profiles values start to disappear and they all when very large messages (1 MByte+) are used.

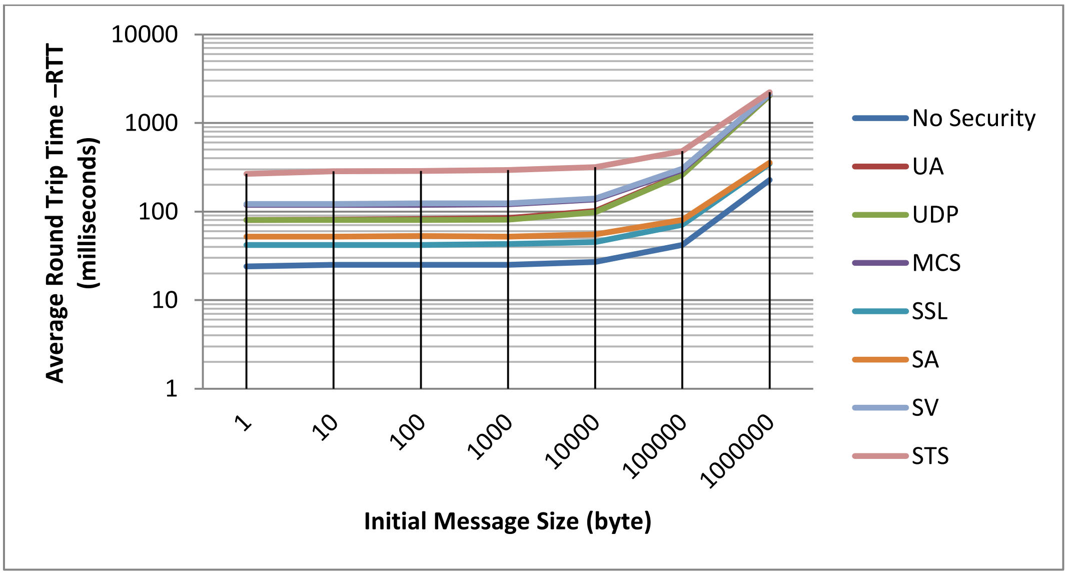

Figure 4-8: Average Round Trip Time –RTT (milliseconds)

The standard deviation values (STDEV) of the round trip time (Figure 4-9) illustrate the variation there is from its normal behaviour, represented by the average. A small STDEV value indicates that the data points tend to be very close to the average, whereas a large value for the STDEV indicates that the data are spread out over a large range of values.

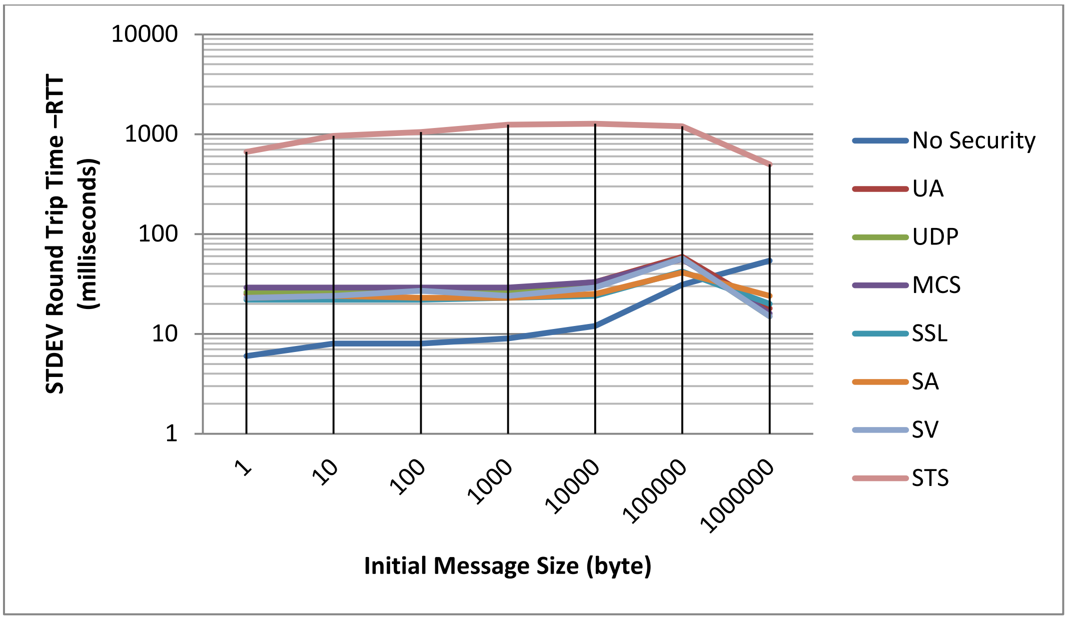

Figure 4-9: STDEV Round Trip Time –RTT (milliseconds)

In small data sizes (1 byte to 1 Kbyte), all the studied profiles, apart from the STS, have a steady STDEV when the data size increases. STS depends on a third party to provide tokens, which means that there are two active communication channels, where the latency could occur in either. This mean that STS based profiles tend to have a variant performance where the round trip time could vary quite often from its normal (average) value. All the other profiles show an increase in the STDEV when the data size increased to 100Kbyte, which then decreases to a smaller value when the data size reaches 1 Mbyte.

Figure 4-10 shows the change in maximum round trip time when the data size increases. The maximum value can be used to describe the “worst case scenario” or the worst performance you can get from a particular Web services. Due to the bigger number of communication channels in the STS profile, this profile has the highest value of maximum RTT. Its maximum values vary slightly in response to changing the data size.

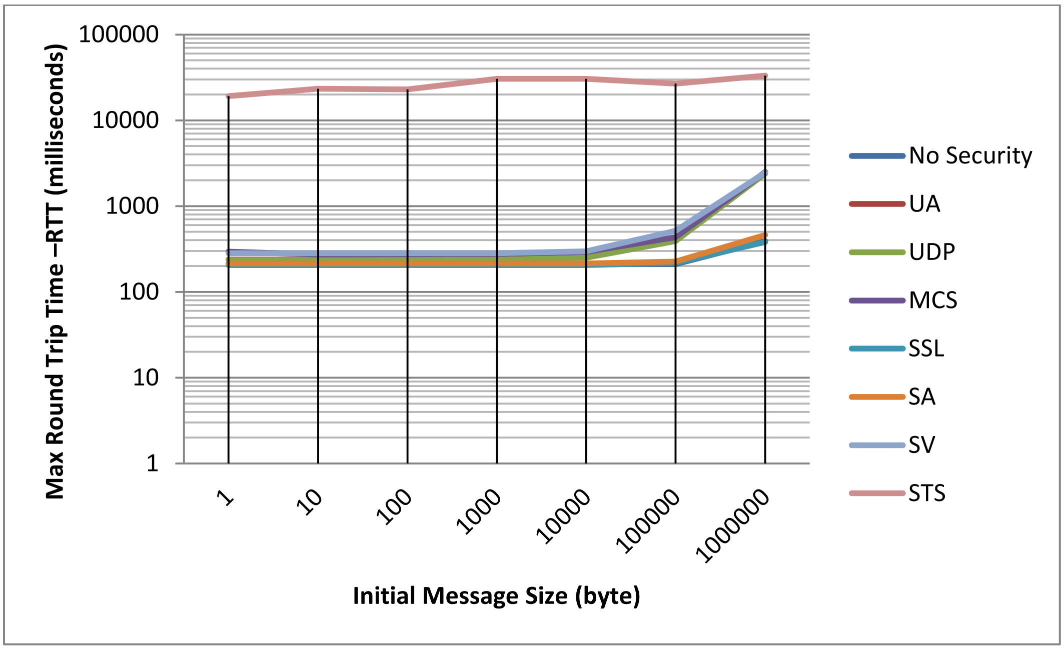

Figure 4-10: Max Round Trip Time –RTT (milliseconds)

On the other hand, all the other profiles have a very similar maximum values in small data size messages (1 byte to 10Kbyte). Significant differences start to appear when the data size reaches 100 Kbyte. The maximum values at the 1 Mbyte data size indicate that the worst case scenarios for transport layer security profiles are still better than those of message layer security profiles. In addition, The STS profile performs significantly worse than the rest, which is different from its behaviour in normal scenarios (average values), where the differences between the performance of all message-level security profiles start to disappear when the message size increases to 1 Mbyte.

4.4 Reliable Messaging

As illustrated in Table 4-4, adding reliable messaging to our Web service resulted in increasing the average response time by around 30 % when applying reliable messaging alone, and more than twice its value when we applied the *Exact Order Delivery* (which drops the throughput to less than half its value).

	Average Response Time (milliseconds)	Max Response Time (milliseconds)	Throughput (response/second)	Average RTT (milliseconds)
Simple (No reliability)	16	32	62.5	107.42
Reliable Messaging (RM)	21	312	47.62	135.11
RM + Order	35	18329	28.57	155.94

Table 4-4: Reliable Messaging Results

The maximum response time recorded in this experiment for Web services with reliable messaging implemented was more than 10 times the maximum value of Web services without reliability assurance. When applying the exact order delivery assurance, the maximum value increased by more than 570%. This is because when a message is lost, the sender endpoint resends the message until its receipt is acknowledged by the receiving endpoint. If these messages are received out of order, the receiving endpoint can rearrange the messages into the correct order. These processes have a huge impact on the performance of Web services, as demonstrated in the previous table.

4.5 Summary

In this chapter, we have compared the performance of several security profiles for Web services. The performance evaluation has shown that profiles that use transport level security are always faster than the message-level security ones. In addition, Message level security protocols have a scalability problem if large messages are exchanged, unlike SSL-based profiles. Within message level security profiles, username authentication based profiles perform better than mutual certificates security. However, the difference is insignificant when using very large size messages. Using digest passwords instead of encrypting the whole username token can slightly improve the performance. Moreover, the performance penalty of using SAML is very small and depends primarily on the underlying security profile. In addition, the performance of STS security profiles is massively less than non-STS profiles and should only be used when the service and its clients are on different domains. Finally, reliable messaging is very important in critical systems because it enables them to overcome the failure of losing messages in transit or delivering them out of order. However, reliability comes at the expense of the performance of Web services, as discussed in the previous section.

Several studies, such as (Shirasuna et al., 2004; Moralis et al., 2007; Novakouski et al, 2010) reached similar conclusions regarding the high performance of SSL when compared to message-layer security mechanisms and the little impact of changing the security token on the performance of Web services. On the other hand, the main difference between our approach and the previous work is that we focused on the performance of security profiles rather than its underlying standards and protocols. We argue that our approach simplifies the security usage and increases the usability when making a decision on which security profile to select for a given application. This is because it shields the developer from the complexity and variety of the different security standards, and allows them to focus on the overall performance of the *easier-to-understand* profiles. Moreover, we studied the impact of changing the message size on the performance of Web services and analysed not only the average values, but we also considered the maximum and standard deviation values in order to reach a better understanding of the overall performance.

This chapter represents the second iteration of the DSR cycle (see section 3.6.2); the results gathered from these experiments provide the basis for the AHP model that is developed in the next chapter. The discussion provided in this chapter may also be used as a performance guideline to be consulted when selecting a security profile for a Web services application.

Chapter 5: Selecting Web Services Security Profiles: A Multi-Criteria Decision Making Approach

5.1 Overview

As discussed in the previous chapter, there are several XML-based security profiles and mechanisms that may be used to satisfy the security requirements of a particular application. The task a Web service developer in an ideal situation is to select the one profile that satisfies all the requirements. On many other occasions, there may be several solutions that satisfy most of the requirements, but not all. This can be considered as a decision problem, where an informed decision should be made by prioritizing the requirements and ranking the available options accordingly to select the profile that matches the most important requirements.

The selection of a candidate solution depends on evaluating the available security profiles against several characteristics derived from the requirements of the given application. A Web services application developer specifies a set of requirements according to the service provider's security preferences and technological constraints, taking into consideration the quality of service expected from the service consumer. The requirements can be divided into several dimensions according to different independent criteria and sub-criteria to create a multi-dimensional model. Each criterion and sub-criterion is given a weight according to its level of importance in the application. Each candidate solution is also weighted against each sub-criterion according to what extent this particular solution matches the requirement represented by this sub-criterion.

Generally, service providers set these weights on the basis of judgment. When there are a small number of selection criteria, then direct judgments may be a possibility. Nevertheless, when the number is large, the judging process may lead to improper selection due to the bias towards key elements only (Godse, Sonar & Mulik, 2008). In such a multi-criteria decision making model, it is important to have quantifiable values that are rational and consistent.

In this chapter, an Analytical Hierarchy Process (AHP) (Saaty, 1982) approach is used to solve the problem of assigning weights to selection features and enable a quantitative basis for selecting a security profile for a Web service. AHP is a well-established method to make these types of decisions. It is a quantitative approach based on relative judgments, with the assurance of consistent output. AHP is able to handle tangible as well as intangible attributes while monitoring the consistency in judgment (Godse, Sonar & Mulik, 2008; Roper-Lowe & Sharp, 1990) .

This chapter also provides a scenario-driven approach to demonstrate situations where the decision making framework is used to make informed choices to rank various service security patterns and select the best possible one to meet the requirements of these scenarios.

Accordingly, the structure of this chapter is as follows: Section 5.2 discusses the decision making concept, while section 5.3 provides an overview of the common methods used in Decision making. Section 5.4 explains the Analytical Hierarchy Process (AHP) and the steps of its implementation. The AHP framework for selecting security profiles for Web services is introduced in section 5.5. The framework is then validated using three different scenarios in Section 5.6. Finally, concluding remarks are presented in section 5.7.

5.2 The Necessity of a Formal Decision Making Framework

The modern enterprise sees itself operating in an ever-complex environment, with technological advances and information overload forcing such enterprises to look and deal with different tasks. The decision making process is one of the most important tasks in any organisation, due to its high-risk implication and future consequences (Saaty, 1982).

In such situations, the multiplicity and complexity of the criteria lead to a complex decision making process. In the real life environment communication links between the members of the decision-making group with a common understanding of the syntax and semantics of the underlying issues are an essential requirement for making an informed decision.

As a consequence of the complexity, stochasticity and the involvement of many decision makers, a disciplined framework for decision making has become a requirement. Thus, many formal decision techniques were developed in the past to tackle these problems. However, these tools were too mathematical or theoretical or only capable of solving problems simpler than the modern ones (Bhushan & Rai, 2004).

On the other hand, advances in the field of the mathematics, operations research, cybernetics, artificial intelligence have been applied and used to develop decision making techniques (Bhushan & Rai, 2004). The underlying principle of these methods is optimisation. This results in a vast expansion of quantitative decision making aid using optimisation techniques.

Decision making is a process of choosing one alternative among a set of alternatives, based on some criteria. This can be achieved by assessing these criteria against each alternative as well as the evaluation of the alternatives based on each criterion. The conclusion of all these evaluations is then used to achieve the relative ranking of the alternatives. In addition, with a group of experts' opinions to be incorporated in the decision making the complication will increase.

Therefore, a structured approach is needed to avoid the ambiguity of the analysis and boost the progress. A generic disciplined process should provide a structure to deal with complex problems, a justification for the decisions, consistency, objectivity and finally the decisions can be repeated and reviewed and are easy to understand.

5.3 Decision Making Methods

There are several tools for solving a decision problem. The selection of an appropriate tool is not an easy task and depends on the concrete decision problem, as well as on the objectives of the decision makers. The chosen approach should employ credible evaluation methods (Baker et al., 2001).

Over decades, several approaches were developed as a try to standardise the process of making decisions. Choosing an appropriate decision making method is dependent on the decision problem type, the attributes of the decision making method and the decision makers' objectives. The use of optimization techniques can also lead to a more deployment of decision making methods (Bhushan & Rai, 2004). The chosen method should thus be justified and evaluated (Baker et al., 2001). In general, the ease of use and the applicability remain an issue for some approaches due to the heavy dependence on Math (too theoretical) or incapability to solve complicated decision problems. For instance, Ranking Approach (RA) (Buss, 1983), non-linear programming model (Badria & Davisb, 2001; Santhanam & Kyparisis, 1996), 0-1goal programming model and Analytical Network Process (ANP) (Lee & Kim, 2000) reliant on complicated mathematical models are difficult to understand and therefor to use.

Numerous multivariate methods also reported to be used ignore decision makers' preferences in the process of decision making (e.g. the Simple Multi-Attribute

Rating technique (SMAR) (Salmeron & Herrero, 2005; Dutta & Burgess, 2003) and Decision Making Units (DMU) (Salmeron & Herrero, 2005)). DMU involves assessing the performance of different units that might be different in their nature such as a computer or a school. The performance is measured considering the amount of inputs involves and outputs generated. The units' performance measures are then compared in a sense that one unit is more efficient that another if it gives more outputs for same inputs quantity or same amount of outputs for smaller set of inputs. This comparison can be represented mathematically by ratio of a sum of outputs over a sum of inputs. Data Envelopment Analysis (DEA) (Salmeron & Herrero, 2005) extends DMU by assigning different weights to outputs and inputs. The weights are different values assigned to make a unit more important than others. DMU and DEA are preferred when there is no need to consider preferences of decision makers as the main intention to compare units' performances.

Pros and Cons Analysis (Baker et al., 2001) could be used for simple decisions with few alternatives and criteria. Kepner-Tregoe (K-T) Decision Analysis (Kepner & Tregoe, 1981) is suitable for fairly complex problems. Multi-Attribute Utility Theory (MAUT) (Edwards & Barron, 1994; Goodwin & Wright, 1999) is a quantitative comparison method used to combine dissimilar measures of costs, risks, and benefits, along with individual preferences, into high-level, aggregated preferences. The Analytical Hierarchy Process (AHP) (Saaty, 1982; Saaty & Vargas, 1984; Saaty & Kearns, 1985; Saaty, 1987; Saaty, 1990a; Saaty, 1990b; Saaty & Vargas, 1991; Saaty & Vargas, 2000; Saaty, 2001; Saaty, 2008) is a quantitative comparison method used to select a preferred alternative by using pair-wise comparisons of the alternatives based on their relative performance against the criteria. The basis of this technique is that humans are more capable of making relative judgments than absolute judgments (Saaty, 2008).

A research conducted by (Kamal, 2008) compared several prioritising approaches and identified AHP as the most-effective one, as shown in Table 5-1.

Characteristics Differentiating the Prioritisation Techniques	**Prioritisation Techniques**				
	AHP	**SMAR**	**DEA**	**RA**	**ANP**
Incorporation of preference structure	✓	–	–	–	–
Synthesised analysis of diverse judgements	✓	–	–	–	–
An intuitive technique	–	–	–	✓	–
Optimising resource allocation for interaction of factors	✓	–	✓	–	✓
Limited attributes to carry out real world decisions	–	✓	✓	✓	✓
Captures individual knowledge and experience	✓	✓	–	–	–
Gives easy understanding of problem situation	✓	–	–	–	✓
Time-consuming process	–	–	–	–	–
Non-linear representation	–	–	–	✓	–
Managing large amount of qualitative/quantitative data	✓	–	–	–	–
Applicability weakened by complex mathematical models	–	–	–	✓	✓
Easy understanding of the prioritisation process	✓	✓	–	✓	–
Quick insight into structure of information	✓	✓	–	–	–
Requires less skill and training	✓	✓	✓	✓	✓
Measure the performance efficiency of decision makers	–	✓	✓	–	–
Structures through symbolic and numeric representation	✓	✓	–	–	–
Supports different viewpoints through rich pictures	✓	–	–	–	–
Techniques not appropriate for all situations	✓	✓	✓	✓	✓
Too much focus on quantifiable calculations	–	✓	✓	✓	✓
Providing a step-wise guideline for prioritising the factors	✓	–	–	–	✓
Accessible data format	✓	–	✓	–	–
Graphical representation	✓	–	–	–	–
Resolves complex problems of choice and prioritisation	✓	–	✓	–	✓

Table 5-1: Comparisons of Decision Making Approaches (Kamal, 2008)

5.4 The Analytical Hierarchy Process (AHP)

The AHP was developed as an organised approach, utilising the experience, intuition and heuristics, to give decision-making the structure of a well-defined methodology derived from sound mathematical principles.

The wide spread AHP acceptance by the decision makers is due to its simplicity and ease of use. AHP helps structure the decision maker’s thoughts and organise the problem in an easy to follow and analyse manner. A wide range of applications has utilised the AHP, including alternative selection (Zeng et al., 2007), resource allocation (Ramanathan, 1995), forecasting (Saaty, 1987; Ülengin, 1994; Jensen, 1982; Jensen & Spencer, 1986), business process re-engineering (Ashayeri, Keij & Bröker, 1998; Wei, Chien & Wang, 2005), quality function deployment (Karsak, Sozer & Alptekin, 2003), balanced scorecard (Ravi, Shankar

& Tiwari, 2005), benchmarking (Lu et al., 1994), public policy decisions (Saaty, 2001), healthcare (Dolan, 1989), multimedia communication (Ghinea, Magoulas & Siamitros, 2005) and many more. Essentially, the AHP has been used to structure the complexity, measurement and synthesis of rankings. These features make it suitable for all these applications and it has been acknowledged as a theoretically sound and market-tested and accepted methodology. AHP's success is given by its almost universal adoption as a new paradigm for decision-making coupled with simplicity of implementation and understanding. Furthermore, its results agree with perceptions and expectations.

The main concepts of the AHP are:

- The AHP is analytic. It assists in analysing the decision problem logically and in establishing numbers based on the decision maker's intuition and feelings which can be validated, questioned and reviewed by others.
- The AHP utilises a hierarchy structure. This property comes naturally with the human tendency to decompose and reduce the complex problems into sub problems to be tackled one by one.
- The AHP defines a step-by-step process for decision making.

These steps will now be described (Saaty & Vargas, 2000):

5.4.1 Hierarchy

The problem is decomposed into a hierarchy of goal, criteria, sub-criteria and alternatives.

Hierarchy indicates a relationship between elements of one level with those of the level immediately below. This relationship percolates down to the lowest levels of the hierarchy and in this manner every element is connected to every other one, at least in an indirect manner.

In the hierarchic structure, at the root of the hierarchy is the goal or objective of the problem being studied and analysed. The leaf nodes are the alternatives to be compared. In between these two levels are various criteria and sub-criteria. It is important to note that when comparing elements at each level a decision-maker has just to compare with respect to the contribution of the lower-level elements to the upper-level one.

5.4.2 Pair-Wise Comparisons

Data are collected from experts or decision-makers corresponding to the hierarchic structure, in the pair-wise comparison of alternatives on a qualitative scale as described below. The comparisons are made for each criterion and converted into quantitative numbers, as illustrated in Table 5-2.

Option	Numerical value(s)
Equal	1
Marginally strong	3
Strong	5
Very strong	7
Extremely strong	9
Intermediate values to reflect fuzzy inputs	2, 4, 6, 8
Reflecting dominance of second alternative compared with the first	Reciprocals

Table 5-2: The Nine-Point Scale For Pair-Wise Comparisons

The pair-wise comparisons of various criteria generated at step 2 are organised into a square matrix. The diagonal elements of the matrix are 1. The criterion in the ith row is better than criterion in the jth column if the value of element (i, j) is more than 1; otherwise the criterion in the jth column is better than that in the ith row. The (j, i) element of the matrix is the reciprocal of the (i, j) element.

$$A=\begin{bmatrix} 1 & \dots & a_{ij} \\ \dots & 1 & \dots \\ 1/a_{ij} & \dots & 1 \end{bmatrix} \quad (2)$$

5.4.3 Eigen Vector

The principal Eigen value and the corresponding normalised right Eigen vector of the comparison matrix give the relative importance of the various criteria being compared. The elements of the normalised eigenvector are termed weights with respect to the criteria or sub-criteria and ratings with respect to the alternatives.

5.4.4 Consistency Ratio

The consistency of the matrix of order n is evaluated. Comparisons made by this method are subjective and the AHP tolerates inconsistency through the amount of redundancy in the approach. If this consistency index fails to reach a required level, then answers to comparisons may be re-examined. The consistency index, CI, is calculated as:

$$CI = (\lambda max - n) / (n - 1) \qquad (3)$$

Where λmax is the maximum eigenvalue of the judgment matrix. This CI can be compared with that of a random matrix, RI (Table 5-3). The ratio derived, CI/RI, is termed the Consistency Ratio (CR). Saaty suggests the value of CR should be less than 10%.

$$CR = \mathrm{CI} / \mathrm{RI} \qquad (4)$$

Order of the matrix	1	2	3	4	5	6	7	8	9	10
Random Consistency Index – RI	0	0	0.58	0.89	1.11	1.25	1.35	1.40	1.45	1.49

Table 5-3: Random Consistency Indices (Saaty, 1990a)

5.4.5 Ratings

The rating of each alternative is multiplied by the weights of the sub-criteria and aggregated to get local ratings with respect to each criterion. The local ratings are then multiplied by the weights of the criteria and aggregated to get global ratings.

5.4.6 Integrating Group Judgments

If the judging process involves multiple experts, then a single consolidated judgment is calculated using a geometric mean to integrate the group judgment.

5.5 The AHP Framework

This section proposes an AHP framework for selecting security profiles for Web services.

5.5.1 Hierarchy

The main goal of this study is to provide a framework that aids Web service developers when they select the best-suited security profile for a certain application. Therefore, we classified the requirements according to three criteria, namely security, configuration and performance.

The security category is subcategorised according to the different security requirements for Web services. The configuration criterion refers to the resources management requirements for the security profiles (certificates, users' database and security service token), as well as the configuration flexibility. The performance criterion is defined by the average round trip time, standard deviation round trip time and maximum round trip time. In this hierarchy model (Figure 5-1), the alternatives are represented by the different profiles that can be used to secure Web services.

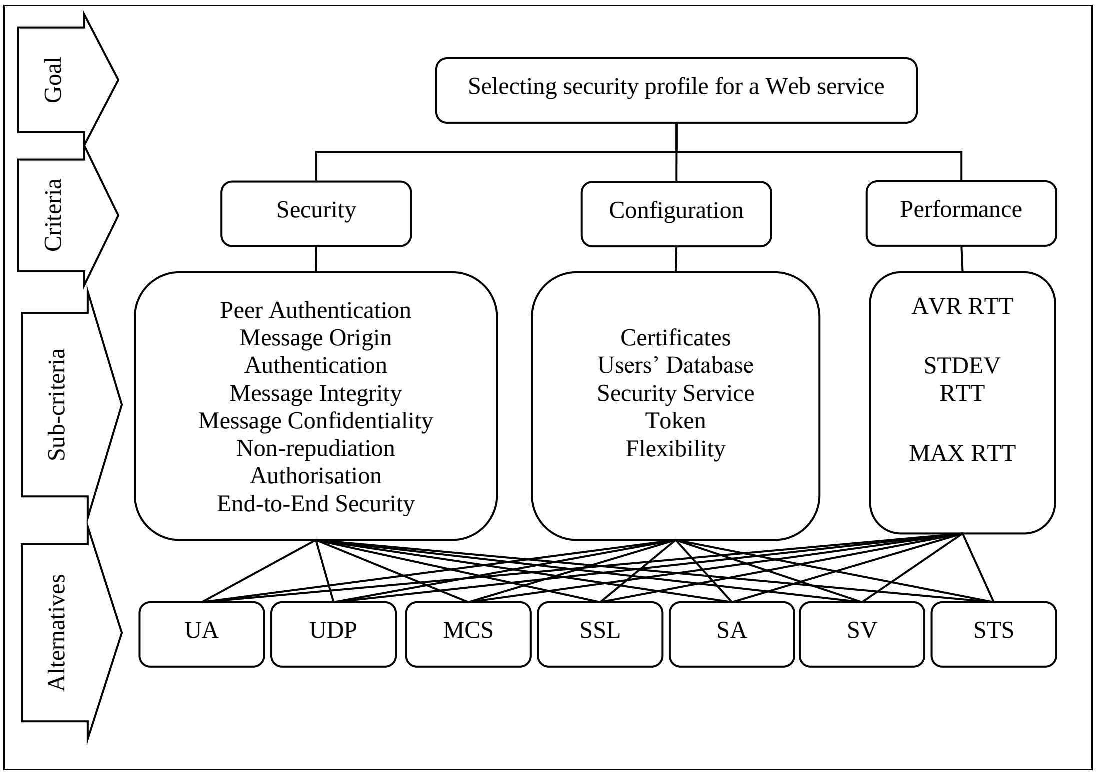

Figure 5-1: An AHP Framework for Web Service Security Profiles

5.5.2 AHP Alternatives : Security Profiles

For the purpose of demonstrating the most common security profiles for Web services, the Metro Web services stack (Sun Microsystems Inc., 2010) is used in this chapter. Metro is an advanced Web services stack that provides interoperability between Java and .Net Web services. The most used feature of Metro is security, which involves streaming encryption/signatures, secure conversation, and trust. To simplify security usage, Metro provides several security profiles that cover the most-used cases. Choosing a profile can be done according to the type of security (transport or message level), type of client credentials (user name/password, X.509 certificate, SAML assertion, Kerberos ticket, or issued token from a third-party) and the role the client credential plays in securing the messages (Carr & Guo, 2009).

In Chapter 2: , several METRO/WSIT security profiles were described and evaluated. Our selection process considered choosing the most common and generic security profiles that represent various security methods, tokens and implementation layers. These profiles are used as alternatives in the proposed AHP model:

- *Username Authentication with Symmetric Key (UA).*
- *Username with digest passwords (UDP).*
- *Mutual Certificates Security (MCS).*
- *Transport Security using SSL (SSL).*
- *SAML Authorisation over SSL (SA).*
- *SAML Sender Vouches with Certificates (SV).*
- *STS Issued Token (STS).*

5.5.3 AHP Criterion 1: Security

There are some major security requirements that must be addressed to ensure the safety of exchanging Web services information through a network (Nakamur, Hada & Neyama, 2002; Geer, 2003). These requirements are:

- *Authentication*: It refers to establishing identity which assures that access to data and applications is limited to those who have appropriate proof of identity. Authentication mechanisms are based on the idea of having a token in the possession of the entity that is authenticated and this token is either software or hardware based. When a Web service starts, information about the authentication status must be carried in the Web service communication. As in standard Web traffic, service provider should authenticate service requesters before sending Web services information. This mechanism is known as peer-to-peer authentication. Another important mechanism is known as message origin authentication. The idea here is that received messages are authenticated based on their origin. This is useful if messages are communicated through a chain of Web services.

- *Integrity*: It refers to the requirement of ensuring that transmitted information has not been changed or modified during transmission. Knowledge about tampering occurrence fulfils integrity requirements. Integrity can be achieved using digital signature, for example.

- *Confidentiality*: It refers to the requirement for exchanged data between two communicating parties not to be available to a third party that may try to pry into the communication. In order to achieve confidentiality, one approach is to use a private connection between the communicating parties, such as a dedicated line or a VPN. However, the critical information of Web services is usually exchanged through untrusted networks, the Internet most likely, where the private connection is not achievable and another approach is used to meet the requirement of confidentiality, which is encryption.

- *Non-repudiation*: It means that the message originator cannot deny sending this message. Any doubt about the message sender throws confidentiality and integrity into question and results could be disastrous. The Web Services Security (WSS) standard assures non-repudiation through its use of the XML Signature standard" (Singhal, Winograd & Scarfone, 2007).

- *Authorisation*: It refers to granting privileges for users and deciding whether an entity is allowed to access particular resources and services or not. Just because a user is authenticated does not mean that they are always authorised. Authorisation software allows administrators to manage a policy for access control to services by giving different privileges to different users and groups. Single sign-on technologies, such as SAML, are used in Web services for both authentication and authorisation.

- *End-to-End Security*: It means that communicated data is signed and encrypted between partners communicating the data throughout the chain.

5.5.4 AHP Criterion 2: Configuration

Each security profile requires configuring some options on the Web service host. These configurations requirements reflect the technological constraints and system preference of the Web service provider. The Web service client may need to be configured as well depending upon the security profile selected by the server side.

For the security profiles mentioned earlier, the regular service configuration requirements (Sun Microsystems Inc., 2010) are:

- *Certificates stores:* there are two types of certificates stores. Keystore is used in the service and client sides to specify the certificates and private keys for the service and client, respectively. Truststore is used in the server side to specify aliases that contain the certificates and trust root of the clients, and vice versa.
- *Security Token Service:* this service implements a protocol that defines message formats and message exchange patterns for issuing, renewing, cancelling, and validating security tokens. The STS we used in our test was deployed in the active mode, as it implements the WS-Trust protocol, as opposed to the WS-Federation passive protocol. The security configuration for the client-side of this application is dependent upon the security mechanism selected for the Security Token Service, and not on the security mechanism selected for the application.
- *Users' database:* to be used by security profiles that require username, and preferably passwords, for authentications.

- Flexibility: The configuration requirements vary in terms of flexibility between the different security profiles. Transport level security profiles are usually easier to configure than the message level ones. Furthermore, some profiles require configuring additional options, such as SAML callback handler on the client side. We use the term “configuration flexibility” in this chapter to refer to how flexible it is to implement a certain security profile.

5.5.5 AHP Criterion 3: Performance

In order to study the performance of the security profiles, in the previous chapter, we used a simple JAX-WS echo application, which consists of a Web service and a client in a dedicated switched network; the client sends different sized messages (from 1 Byte to 1MByte) and the Web service echoes (sends back) the same message received. Every security mechanism has been tested under the default settings of its profile to secure the request and response SOAP messages, where all the tested profiles provide peer authentication (Sun Microsystems Inc., 2010).

We measured the time spent in requesting and responding on the client side as round trip time (RTT), which starts from the moment the client starts initializing a request to the server until receiving the final response from the server. This includes for example the time needed to authenticate and obtain an assertion from an Identity Provider in SAML-based profiles, or an issued token from a Security Token Service (STS) in STS-based profiles. However, some security profiles, such as MCS, SSL and SV, require an out-of-band exchange for the digital certificates. The method and frequency of exchanging the certificates are not part of the default settings of these profiles; hence the time needed to exchange them can vary. In order to test the performance of such profiles, we assumed that the digital certificates already exist on the server as well as the client, and flagged the *Certificates Stores* sub-criterion in the AHP model to reflect the dependency on these certificates as a configuration issue.

Accordingly, the considered performance parameters are: (1) *average RTT*, (2) *maximum RTT*, and (3) *standard deviation RTT*.

5.5.6 Data Collection

In order to evaluate the performance of the different security profiles, a performance test was conducted on various security profiles applied individually on a simple Web service and tested every time with different initial message sizes. In Chapter 4: the results were used as basis for the comparison between the security profiles in terms of performance.

When selecting a security profile to secure a Web service, it is important to consider the initial data size exchanged between the service provider and client. In the previous chapter, we demonstrated that some security profiles perform differently when changing the size of the exchanged data. Therefore, it is important for the service developer to estimate or measure the expected size of the exchanged data when selecting a security profile.

In this framework, we used the performance measurements of data sizes that range from 1 byte and 1 Megabyte. A developer would choose an appropriate selection window for the application, and then, the results are aggregated accordingly. The Factor Analysis dimension reduction method (FA) (Kim & Mueller, 1993) was used to reduce the number of measurements into a single value that describes the performance measurements of the selected window for each security profile. Since the number of dimensions (i.e. data size categories) is relatively small, medians and geometric means can be biased (Li, Hastie & Church, 2007).

The representatives resulted were then normalised to cover the scale from 1 to 9 to be used in the pair-wise comparison step. The security profile with the largest FA value for a selected window of data sizes should have a normalised value of 1 (lowest) as this profile has the longest round trip time. Similarly, the security profile with the smallest FA value should have the value of 9 (highest) to indicate that this profile has the best performance, or the shortest round trip time.

In this chapter, the AHP framework used a data size window that covered the whole range (1 byte to 1 Megabyte). Table 5-4, Table 5-5 and Table 5-6 illustrate results for the average, standard deviation and maximum RTT, respectively.

Security Profile	Initial Message Size (Byte)							FA Value	Normalised Value
	1	10	100	1 K	10 K	100 K	1 M		
UA	80	81	82	84	101	263	2055	-0.04871	6.798309
UDP	80	80	80	81	98	259	2051	-0.06932	6.854555
MCS	119	119	119	121	137	295	2109	0.33679	5.746259
SSL	42	42	42	43	45	71	348	-0.85547	9
SA	52	52	53	52	55	80	357	-0.75276	8.719699
SV	122	122	124	124	140	303	2110	0.37639	5.638189
STS	266	285	288	294	316	481	2229	2.07595	1

Table 5-4: Factor Analysis and Normalised Values (Average RTT Results)

Security Profile	Initial Message Size (Byte)							FA Value	Normalised Value
	1	10	100	1 K	10 K	100 K	1 M		
UA	25	28	28	28	33	59	18	-0.34493	8.955704
UDP	26	26	27	27	31	57	15	-0.34978	8.969387
MCS	29	29	29	29	33	56	16	-0.344	8.95308
SSL	22	22	22	23	24	42	20	-0.36063	9
SA	23	24	23	23	25	41	24	-0.35538	8.985187
SV	23	24	27	24	29	57	15	-0.35418	8.981802
STS	664	957	1048	1246	1272	1201	498	2.47481	1

Table 5-5: Factor Analysis and Normalised Values (STDEV RTT Results)

Security Profile	Initial Message Size (Byte)							FA Value	Normalised Value
	1	10	100	1 K	10 K	100 K	1 M		
UA	237	237	236	239	253	400	2509	-0.32987	8.838517
UDP	235	235	235	237	251	393	2398	-0.33284	8.846821
MCS	295	274	275	277	291	435	2489	-0.32621	8.828285
SSL	205	205	205	206	205	218	381	-0.38763	9
SA	215	214	215	215	214	226	460	-0.38478	8.992032
SV	282	282	282	282	298	516	2464	-0.32544	8.826132
STS	19218	23250	22973	30351	30366	26838	33229	2.47385	1

Table 5-6: Factor Analysis and Normalised Values (Max RTT results)

5.5.7 Pair-wise Comparisons Matrices for Alternatives

In this model, there are 14 pair-wise comparison matrices for the seven alternatives with respect to all the sub-criteria connected with the alternatives. Due to space limitations and to avoid repetition, a few representative matrices are shown in this section. However, all the matrices are provided in Appendix B.

For each sub-criterion within the security criteria, we rely on judgments derived from the literature study. The words that have been used in literature to describe the strength of alternatives with respect to a specific sub-criterion are used to compare these alternatives. The comparisons then are converted to numerical values using the nine-point scale for pair-wise comparisons (Table 5-2).

For example, in terms of peer authentication, security mechanisms that use asymmetric key cryptography are considered marginally stronger than those that use username tokens (Sun Microsystems Inc., 2010). Hence, MCS is 3 times more than UA in a 9-point scale, as shown in Table 5-7. Although all the examined security profiles guarantee peer-authentication, digital certificates used in the asymmetric key methods (MCS, SSL, SV, etc…) are usually stronger in establishing identity than username/password authentication methods (UA and UDP), which can be sometimes compromised.

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	1/3	1/3	1/3	1/3	1/3
UDP	1	1	1/3	1/3	1/3	1/3	1/3
MCS	3	3	1	1	1	1	1
SSL	3	3	1	1	1	1	1
SA	3	3	1	1	1	1	1
SV	3	3	1	1	1	1	1
STS	3	3	1	1	1	1	1

Table 5-7: Pair-wise Comparison Matrix for the Alternatives with Respect to Peer Authentication

The configuration elements (sub-criteria) are also based on judgments. For instance, we compared the alternatives with respect to the configuration flexibility; assuming that implementing the transport layer security profile (SSL) is the easiest and most straightforward way. Some profiles require configuring additional options, such as the username based profiles (UA and UDP) which require some configurations to the keystore on the server-side and the trust-store on the client-side. Whilst SAML-based profiles require configuring callback handler on the client side, STS-based profiles require configuring the security token service node, in addition to the service provider and client nodes, which make them the most difficult to implement. Table 5-8 shows the values of the pair-wise judgments with respect to flexibility.

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	2	1/2	2	2	6
UDP	1	1	2	1/2	2	2	6
MCS	1/2	1/2	1	1/3	1	1	5
SSL	2	2	3	1	3	3	7
SA	1/2	1/2	1	1/3	1	1	5
SV	1/2	1/2	1	1/3	1	1	5
STS	1/6	1/6	1/5	1/7	1/5	1/5	1

Table 5-8: Pair-wise Comparison Matrix for the Alternatives with Respect to Flexibility

On the other hand, the performance of the different security profiles can be measured as described in the previous section. Thus, the pair-wise comparisons here are done based on actual data rather than judgments, as illustrated in Table 5-9.

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	0.99	1.18	0.76	0.78	1.21	6.8
UDP	1.01	1	1.19	0.76	0.79	1.22	6.85
MCS	0.85	0.84	1	0.64	0.66	1.02	5.75
SSL	1.32	1.31	1.57	1	1.03	1.6	9
SA	1.28	1.27	1.52	0.97	1	1.55	8.72
SV	0.83	0.82	0.98	0.63	0.65	1	5.64
STS	0.15	0.15	0.17	0.11	0.11	0.18	1

Table 5-9: Pair-wise Comparison Matrix for the Alternatives with Respect to Average RTT

5.5.8 Raised Power Matrices

All the comparison matrices should be raised to a higher power to improve accuracy (Saaty, 2008). Table 5-10 shows a matrix derived from the comparison matrix of the flexibility by squaring it twice.

	UA	UDP	MCS	SSL	SA	SV	STS
UA	357.15	357.15	635.35	222.16	635.35	635.35	2522.64
UDP	357.15	357.15	635.35	222.16	635.35	635.35	2522.64
MCS	200.48	200.48	356.82	124.67	356.82	356.82	1416.79
SSL	591.91	591.91	1052.84	368.37	1052.84	1052.84	4182.67
SA	200.48	200.48	356.82	124.67	356.82	356.82	1416.79
SV	200.48	200.48	356.82	124.67	356.82	356.82	1416.79
STS	53.15	53.15	94.6	33.08	94.6	94.6	376.07

Table 5-10: Raised Power Matrix (Flexibility)

5.5.9 Normalised Matrix and Eigen Vectors

To normalise the previous matrices, each element is divided by the sum of its column. The Eigen vector is then calculated by dividing the sum of each row of the normalised matrix by the number of its elements. A representative normalised matrix of the matrix in Table 5-8 and its Eigen vector are shown in Table 5-11, while Table 5-12 illustrates the rankings of the alternatives against each sub-criterion.

The consistency ratio for this matrix is CI/RI = 1.12% < 10%, so the weights are accepted.

	UA	UDP	MCS	SSL	SA	SV	STS	Eigen Vector
UA	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.1821
UDP	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.1821
MCS	0.10	0.10	0.10	0.10	0.1	0.10	0.10	0.1023
SSL	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.3019
SA	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.1023
SV	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.1023
STS	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.0271

Table 5-11: Normalised Matrix and Eigen Vector (Flexibility)

Criteria	Sub-criteria	Security Profiles						
		UA	UDP	MCS	SSL	SA	SV	STS
Security	Peer Authentication	0.0588	0.0588	0.1765	0.1765	0.1765	0.1765	0.1765
	Message Origin Authentication	0.2308	0.2308	0.2308	0.0256	0.0256	0.0256	0.2308
	Message Integrity	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429
	Message Confidentiality	0.0588	0.0588	0.1765	0.1765	0.1765	0.1765	0.1765
	Non-repudiation	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429
	Authorisation	0.0323	0.0323	0.0323	0.0323	0.2903	0.2903	0.2903
	End-to-End Security	0.1915	0.1915	0.1915	0.0213	0.0213	0.1915	0.1915
Configuration	Certificates Exchange	0.0435	0.0435	0.3913	0.3913	0.0435	0.3913	0.0435
	Users Database	0.2903	0.2903	0.0323	0.2903	0.0323	0.0323	0.0323
	Security Service Token	0.0667	0.0667	0.0667	0.0667	0.0667	0.0667	0.6000
	Flexibility	0.1821	0.1821	0.1023	0.3019	0.1023	0.1023	0.0271
Performance	AVR RTT	0.1554	0.1567	0.1313	0.2057	0.1993	0.1289	0.0229
	STD RTT	0.1633	0.1635	0.1632	0.1641	0.1638	0.1638	0.0182
	MAX RTT	0.1627	0.1628	0.1625	0.1656	0.1655	0.1624	0.0184

Table 5-12: Ratings for the Alternatives on Each Criterion

5.6 Scenarios

In this section we illustrate the usage of our proposed framework by adopting three common scenarios (Microsoft Corporation, 2005) used by Microsoft to demonstrate the different Web service security considerations and solutions in common Web services interactions. Additionally, we augmented these scenarios to include performance and configuration requirements besides the security requirements provided by the original solution.

Each scenario starts with a high-level description of the application followed by an identification of requirements and preferences for the application.

5.6.1 Public Web Service Scenario (S1)

Description:

A distributor uses Web services to provide catalogue information to online sellers that provide online shopping services. The sellers access the Web service from their Web applications to display current items available from the distributor. The Web service provider has the following requirements:

Web services clients require direct access to the Web service. Sellers accessing the Web service must be authenticated, and data passed between the service and clients contains some information, such as account information, that must be protected while in transit. The Web service provider has a database for the sellers that are allowed to use this service. A high performance is expected by the clients as they should get instant responses when using the service.

Solution Factors:

The following factors have to be considered for the distributer we service:

- Merchant accounts are stored in a custom database or directory service.
- The message data must be protected during transit.
- Performance must be considered.

AHP Application

According to the previous description, the three criteria are compared against each other's. Within each criterion, the sub-criteria are also pair-compared in terms of their importance to compose the pair-wise comparison matrix. In this scenario we assumed the performance should have the highest priority since we are dealing with an online application. It is *strongly more important* than configuration. Security is *marginally less important* in this application than the performance. The matrix is then raised to a higher power and normalised to calculate the Eigen vector (Priorities), see Table 5-13.

	Security	Configuration	Performance	Priorities
Security	1	3	1/3	0.2583
Configuration	1/3	1	1/5	0.1047
Performance	3	5	1	0.6370
Consistency Ratio = 3.32 %				

Table 5-13: Pair-wise Comparison Matrix for the Main Criteria with Respect to the Goal (S1)

With respect to security criteria, each sub-criterion that satisfies a requirement is considered to be *extremely more important* than a sub-criterion that does not represent a requirement in the application (Table 5-14). The security requirements that should be fulfilled in this application are: peer authentication, integrity, confidentiality and non-repudiation, while the support of message origin authentication, authorisation and end-to-end security is not required in this particular application.

	Peer Authentication	Message Origin Authentication	Message Integrity	Message Confidentiality	Non-repudiation	Authorisation	End-to-End Security	Local weight
Peer Authentication	1	9	1	1	1	9	9	0.2308
Message Origin Authentication	1/9	1	1/9	1/9	1/9	1	1	0.0256
Message Integrity	1	9	1	1	1	9	9	0.2308
Message Confidentiality	1	9	1	1	1	9	9	0.2308
Non-repudiation	1	9	1	1	1	9	9	0.2308
Authorisation	1/9	1	1/9	1/9	1/9	1	1	0.0256
End-to-End Security	1/9	1	1/9	1/9	1/9	1	1	0.0256
Consistency Ratio = 0 %								

Table 5-14: Pair-wise Comparison Matrix for the Sub-criteria with Respect to the Security (S1)

Within the configuration criteria, based on the idea that the service provider has a database for the names of its clients, the user database management option is *extremely more important* than managing exchanged certificates out-of-band. All the Web clients are accessing the service directly, where there is *no need* to enable the management of security service token. Table 5-15 shows the comparisons.

	Certificates	Users DB	Security Token Service	Flexibility	Local weight
Certificates	1	1/9	1	1/7	0.0522
Users DB	9	1	9	3	0.5953
Security Token Service	1	1/9	1	1/7	0.0522
Flexibility	7	1/3	7	1	0.3004
Consistency Ratio = 3.41 %					

Table 5-15: Pair-wise Comparison Matrix for the Sub-criteria with Respect to the Configuration (S1)

For the performance criteria, we presume that the average value of the round trip time has *stronger importance* than the standard deviation value, and *very stronger importance* than the maximum value, as shown in Table 5-16.

	AVR RTT	STDEV RTT	MAX RTT	Local weight
AVR RTT	1	5	7	0.7306
STDEV RTT	1/5	1	3	0.1884
MAX RTT	1/7	1/3	1	0.0810
Consistency Ratio = 5.59 %				

Table 5-16: Pair-wise Comparison Matrix for the Sub-criteria with Respect to the Performance (S1)

The next step is to calculate the global weights vector by multiplying the local weight of each sub-criterion by the priority of its criterion, as illustrated in Table 5-17.

Criteria	Priorities	Subcriteria	Local weight	Global Weight Criteria x subcriteria
Security	0.2583	Peer Authentication	0.2308	0.0596
		Message Origin Authentication	0.0256	0.0066
		Message Integrity	0.2308	0.0596
		Message Confidentiality	0.2308	0.0596
		Non-repudiation	0.2308	0.0596
		Authorisation	0.0256	0.0066
		End-to-End Security	0.0256	0.0066
Configuration	0.1047	Certificates Exchange	0.0522	0.0055
		Users Database	0.5953	0.0623
		Security Service Token	0.0522	0.0055
		Flexibility	0.3004	0.0315
Performance	0.6370	AVR RTT	0.7306	0.4654
		STD RTT	0.1884	0.1200
		MAX RTT	0.0810	0.0516

Table 5-17: Local and Global Weights (S1)

The results are synthesised by multiplying each alternative weights vector by the global weights vector. The resulting weights are added for each alternative to calculate its final priority, as shown in Table 5-18.

Finally, the results can also be presented in the ideal form by dividing each priority by the maximum priority to make this first ranked alternative an ideal one with the others getting their proportionate value, as demonstrated in Table 5-19.

The AHP framework suggests the usage of the transport security using SSL to secure the service because of the tendency towards a high performance in the given scenario. Transport layer security mechanisms are normally faster than message layer security profiles and easier to configure. Although this profile is a point-to-point security mechanism, it is enough to satisfy the security requirements of the proposed application. Microsoft (Microsoft Corporation, 2005) suggests using Username Token and HTTPs in this scenario, which is an equivalent to the Transport Security using SSL provided by the Metro Web service Stack (Sun Microsystems Inc., 2010).

Criteria	Sub-criteria	Security Profiles						
		UA	UDP	MCS	SSL	SA	SV	STS
Security	Peer Authentication	0.0035	0.0035	0.0105	0.0105	0.0105	0.0105	0.0105
	Message Origin Authentication	0.0015	0.0015	0.0015	0.0002	0.0002	0.0002	0.0015
	Message Integrity	0.0085	0.0085	0.0085	0.0085	0.0085	0.0085	0.0085
	Message Confidentiality	0.0035	0.0035	0.0105	0.0105	0.0105	0.0105	0.0105
	Non-repudiation	0.0085	0.0085	0.0085	0.0085	0.0085	0.0085	0.0085
	Authorisation	0.0002	0.0002	0.0002	0.0002	0.0019	0.0019	0.0019
	End-to-End Security	0.0013	0.0013	0.0013	0.0001	0.0001	0.0013	0.0013
Configuration	Certificates Exchange	0.0002	0.0002	0.0021	0.0021	0.0002	0.0021	0.0002
	Users Database	0.0181	0.0181	0.002	0.0181	0.002	0.002	0.002
	Security Service Token	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0033
	Flexibility	0.0057	0.0057	0.0032	0.0095	0.0032	0.0032	0.0009
Performance	AVR RTT	0.0723	0.0729	0.0611	0.0957	0.0927	0.06	0.0106
	STD RTT	0.0196	0.0196	0.0196	0.0197	0.0197	0.0197	0.0022
	MAX RTT	0.0084	0.0084	0.0084	0.0085	0.0085	0.0084	0.0009
Total Priority		0.1517	0.1523	0.1378	0.1925	0.1669	0.1372	0.0628
Ranking		4	3	5	1	2	6	7

Table 5-18: Synthesizing to Obtain Final Results and Ranking (S1)

Security Profile	Normalised Priorities	Idealised Priorities	Ranking
UA	0.1517	0.7959	4
UDP	0.1523	0.7991	3
MCS	0.1378	0.7230	5
SSL	0.1925	1.0000	1
SA	0.1669	0.8757	2
SV	0.1372	0.7198	6
STS	0.0628	0.3295	7

Table 5-19: Final Results shown as Normalised and Idealised Priorities (S1)

In addition, our AHP framework provides an order for the other possible alternatives. In this scenario, peer-to-peer authentication and configuration, Transport layer based security profiles are the top of the ranking.

5.6.2 Intranet Web Service Scenario (S2)

Description:

A company uses an internal application to access operations provided by a Web services. Mutual authentication is required for all Web services interactions. The application must support single sign on (SSO) capabilities. Message data are sensitive and must be protected against unauthorised access and the message must not be tampered with during transit.

Solution Factors:

The following factors have to be considered for the distributer Web service:

- Mutual authentication is required for all Web service interactions.
- Applications must support single sign on (SSO) capabilities.
- Message data is sensitive and must be protected against unauthorised access.
- The message must not be tampered with during transit.

AHP Application

The pair-wise comparison matrices and results are shown below:

	Security	Configuration	Performance	Priorities
Security	1	5	3	0.6370
Configuration	1/5	1	1/3	0.1047
Performance	1/3	3	1	0.2583
Consistency Ratio = 3.32 %				

Table 5-20: Pair-wise Comparison Matrix for the Main Criteria with Respect to the Goal (S2)

	Peer Authentication	Message Origin Authentication	Message Integrity	Message Confidentiality	Non-repudiation	Authorisation	End-to-End Security	Local weight
Peer Authentication	1	3	3	3	3	1	1	0.2308
Message Origin Authentication	1/3	1	1	1	1	1/3	1/3	0.0769
Message Integrity	1/3	1	1	1	1	1/3	1/3	0.0769
Message Confidentiality	1/3	1	1	1	1	1/3	1/3	0.0769
Non-repudiation	1/1	1	1	1	1	1/1	1/1	0.0769
Authorisation	1	3	3	3	3	1	1	0.2308
End-to-End Security	1	3	3	3	3	1	1	0.2308
Consistency Ratio = 0 %								

Table 5-21: Pair-wise Comparison Matrix for the Sub-criteria with Respect to the Security (S2)

	Certificates	Users DB	Security Token Service	Flexibility	Local weight
Certificates	1	9	9	5	0.6693
Users DB	1/9	1	1	1/5	0.0555
Security Token Service	1/9	1	1	1/5	0.0555
Flexibility	1/5	5	5	1	0.2198
Consistency Ratio = 4.99 %					

Table 5-22: Pair-wise Comparison Matrix for the Sub-criteria with Respect to the Configuration (S2)

	AVR RTT	STDEV RTT	MAX RTT	Local weight
AVR RTT	1	5	7	0.7306
STDEV RTT	1/5	1	3	0.1884
MAX RTT	1/7	1/3	1	0.0810
Consistency Ratio = 5.59 %				

Table 5-23: Pair-wise Comparison Matrix for the Sub-criteria with Respect to the Performance (S2)

Security Profile	Normalised Priorities	Idealised Priorities	Ranking
UA	0.1197	0.6411	7
UDP	0.1199	0.6422	6
MCS	0.1593	0.8532	2
SSL	0.1200	0.6427	5
SA	0.1507	0.8072	3
SV	0.1867	1.0000	1
STS	0.1436	0.7691	4

Table 5-24: Final Results shown as Normalised and Idealised Priorities (S2)

The AHP framework results indicate that using SAML Sender Vouches with Certificates (SV) is the most appropriate alternative for this kind of applications. This profile assures the fulfillment of the high security requirement due to the use of mutual certificates for integrity and confidentiality, while ensuring a good performance. In addition, the SV profile satisfies the requirement of providing single sign on capability by using SAML tokens for authorisation.

5.6.3 Multiple Internet Web Service Scenario (S3)

Description:

A travel booking franchise provides a Web application that travel agents can use to search for and book travel packages. The Web application uses several Web services to perform the operations of searching for and booking packages. The travel booking Web application is accessible from the Internet. However, only the Web application can access the Web services that the application calls. Each Web service has an independent data store.

The travel booking application has the following features:

- Travel agents in a travel franchise help customers to book tour packages.
- Two Web services are used: a travel packages Web service, and an online booking Web service.

- The travel packages Web service provides travel product catalogue information such as tour dates, itineraries, and prices.
- The online booking Web service allows travel agents to book tour packages on behalf of the customers.
- Identity propagation is needed for the online booking Web service because the database needs to keep a record of each travel agent who makes a travel request. Customers can go to any travel agent in the franchise to book a tour.
- During peak travel seasons, user activity is high. This means that performance must be considered.

Solution Factors:

The following factors have to be considered for the distributer Web service:

- Travel agent user accounts are stored in a database.
- Mutual authentication is required.
- SSO support is required.
- Performance must be considered.
- Sensitive data must be protected against unauthorised access.
- Web services are behind a firewall.

AHP Application

The pair-wise comparison matrices and results are illustrated in the tables below.

	Security	Configuration	Performance	Priorities
Security	1	5	7	0.7306
Configuration	1/5	1	3	0.1884
Performance	1/7	1/3	1	0.0810
Consistency Ratio = 5.59 %				

Table 5-25: Pair-wise Comparison Matrix for the Main Criteria with Respect to the Goal (S3)

	Peer Authentication	Message Origin Authentication	Message Integrity	Message Confidentiality	Non-repudiation	Authorisation	End-to-End Security	Local weight
Peer Authentication	1	1	1	1	1	1	1	1
Message Origin Authentication	1	1	1	1	1	1	1	1
Message Integrity	1	1	1	1	1	1	1	1
Message Confidentiality	1	1	1	1	1	1	1	1
Non-repudiation	1	1	1	1	1	1	1	1
Authorisation	1	1	1	1	1	1	1	1
End-to-End Security	1	1	1	1	1	1	1	1
Consistency Ratio = 0 %								

Table 5-26: Pair-wise Comparison Matrix for the Sub-criteria with Respect to the Security (S3)

	Certificates	Users DB	Security Token Service	Flexibility	Local weight
Certificates	1	1	1/9	1/3	0.0630
Users DB	1	1	1/9	1/3	0.0630
Security Token Service	9	9	1	7	0.7186
Flexibility	3	3	1/7	1	0.1554
Consistency Ratio = 3.41%					

Table 5-27: Pair-wise Comparison Matrix for the Sub-criteria with Respect to the Configuration (S3)

	AVR RTT	STDEV RTT	MAX RTT	Local weight
AVR RTT	1	5	7	0.7306
STDEV RTT	1/5	1	3	0.1884
MAX RTT	1/7	1/3	1	0.0810
Consistency Ratio = 5.59 %				

Table 5-28: Pair-wise Comparison Matrix for the Sub-criteria with Respect to the Performance (S3)

Security Profile	Normalised Priorities	Idealised Priorities	Ranking
UA	0.1206	0.5343	6
UDP	0.1207	0.5347	5
MCS	0.1425	0.6312	3
SSL	0.1125	0.4984	7
SA	0.1302	0.5766	4
SV	0.1479	0.6551	2
STS	0.2257	1.0000	1

Table 5-29: Final Results shown as Normalised and Idealised Priorities (S3)

The results recommend using single Security Token Service (STS) to establish a chain of trust between the server and the client; especially that service providers and clients are in different managed environments, each with its own resources, where confidentiality is a major issue. As a confirmation of this recommendation, the Sun Microsystems Metro User Guide (Sun Microsystems Inc., 2010) and Microsoft Patterns and Practices Document (Microsoft Corporation, 2005) indicate using an STS-based security profile for this kind of applications.

5.7 Summary

Selecting the best possible security profile for a Web service in a given application can be a difficult task for Web services developers, as there are many different options available to help secure Web services. The selection process is further complicated by the fact that different systems and projects can have different requirements and specifications that drive their security decisions. Furthermore, securing Web services tends to aggravate the performance of these services. Therefore, the developer is expected to select a security profile that satisfies not only the security requirements, but also the performance expectations.

In this chapter, we have provided a novel multi-criteria decision making framework based on the analytical hierarchy process in order to help developers prioritise their security, performance and configuration requirements in order to rank the available alternatives according to their suitability for a specified application. The data used in building this model has been derived from a literature study as well as actual performance testing.

While other models (Wu & Chang, 2007; Zuo, Wang & Wu, 2008; Godse, Sonar & Mulik, 2008; Casola et al., 2009; Thirumaran et al.; 2011) have focused on the service consumer point of view to address the Web services composition problem, this chapter has its focus on the service developer viewpoint during the development process. We have also provided common security usage scenarios for Web services and applied the proposed framework on them to test its validity. The framework results match the security recommendations for these scenarios.

In order to evaluate the applicability of the proposed Multi-Criteria Decision Making framework in practical settings, we have adopted three common usage scenarios for Web services security, as presented by Microsoft and Sun. Comparing the results of the MCDM framework to the suggestions of Microsoft and Sun, and reaching similar conclusions illustrate the validity of the model. However, there are other cases where a Web service developer may not be able to rely solely on documentations of best practice, such as:

1) New or unusual scenarios, where no or very little documentations of best practice are available.

2) More than one recommended solution can be implemented.

3) The recommended solution cannot be implemented due to system/configurations limitations.

In addition, our MCDM includes not only the usual security factors, but also configuration and performance considerations. It rates and provides an order for all the possible alternatives that can be considered by a Web services developer, so he/she can select which profile is best to be implemented.

The proposed framework can be implemented as a separate Web service that Web developers can consult when selecting security profiles for their Web services. Alternatively, the MCDM model can be extended to cover other quality dimensions and therefore act as an additional sub-layer that sits at the top of the Quality of Service (QoS) layer in the Web services protocol stack, as illustrated in Figure 5-2.

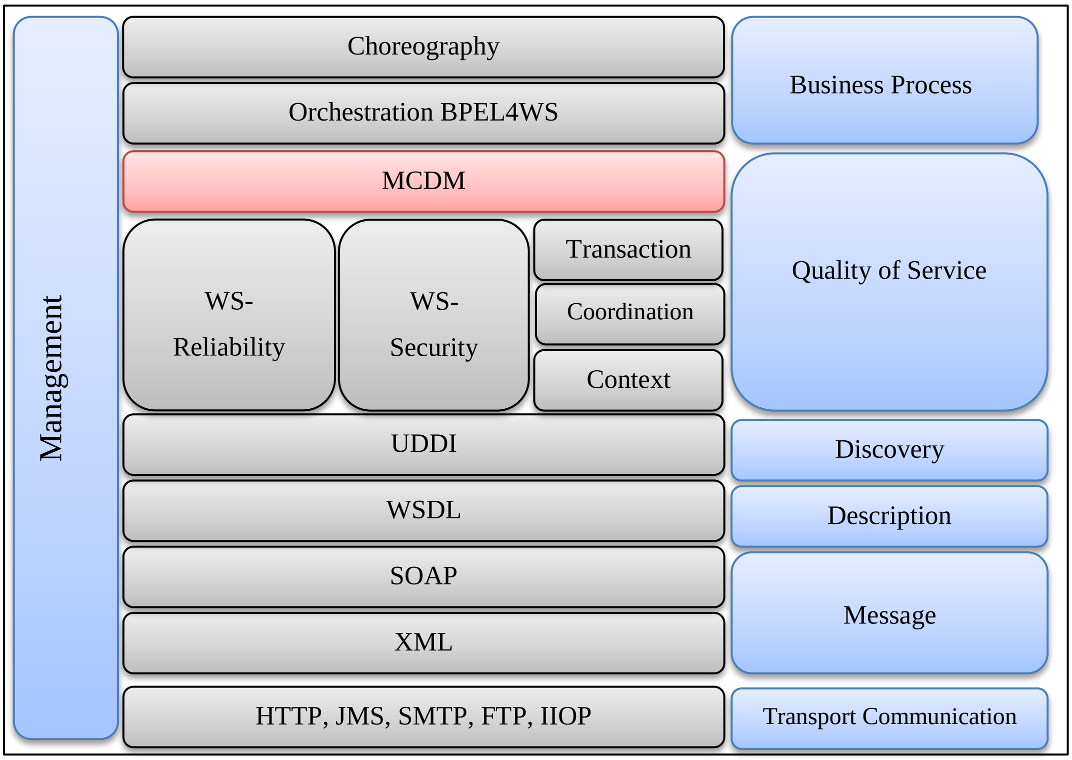

Figure 5-2: The MCDM in the Web Services Stack

This position enables the MCDM model to rely on the underlying QoS protocols as they represent the criteria/alternatives, and keeps it independent of the upper business logic layer and the lower discovery, description, message and transport layers.

In Appendix C, We present the AHP framework as a generic Java class to perform the AHP calculations and a Java Main class to run the AHP tool. The framework and the tool are the outcomes of the third iteration of the DSR cycle (see section 3.6.3)

Chapter 6: Information Hiding in SOAP Messages: A Steganographic Method for Web Services

6.1 Overview

Digital steganography is the art and science of hiding communications; a steganographic system thus embeds secret data in public cover media so as not to arouse an eavesdropper's suspicion. There are still very limited methods of steganography to be used with communication protocols, which represent unconventional but promising steganography mediums. In this chapter, we discuss and analyse a number of steganographic studies in text, XML as well as SOAP messages. Then, we propose a novel steganography method to be used for SOAP messages within Web services environments. The method is based on rearranging the order of specific XML elements according to a secret message. This method has a high imperceptibility; it leaves almost no trail because of using the communication protocol as a cover medium, and since it keeps the structure and size of the SOAP message intact. The method is empirically validated using a feasible scenario so as to indicate its utility and value.

6.2 Steganography vs. Encryption

Secure and secret communication methods are needed for transmitting messages over the Internet. Cryptography scrambles the message so that it cannot be understood. However, it makes the message suspicious enough to attract eavesdropper's attention. Additionally, due to the increase of computers capabilities and cipher texts availability, cryptographic techniques could be vulnerable. However, this vulnerability can be reduced significantly using steganography, which is a method of covert communication and information security.

Unlike encryption, steganography hides even the existence of secret information rather than hiding its meaning only. Thus, steganography is the art of hiding secret messages within other innocuous-looking cover files (i.e. images, audio, video, and text files) so that it cannot be observed. Consequently, steganography aims to hide the very existence of communication by embedding messages within other cover objects. As a result, the purpose of steganography is to keep others from thinking that a secret message even exists within the stego files.

Using only encryption for secret communication draws the attention of others. Therefore, steganography combined with cryptography raises the security level and would be the most secure method to go.

Steganography can be considered as a solution to exchange secret information and news between people around the world over the Internet without any fear of the message being detected.

Related to steganography but distinct from it, watermarking is a data hiding technique that protects digital documents, files, or images against removal of copyright information. Therefore, the goal of steganography is the secret messages while the goal of watermarking is the cover object itself (Venkatraman, Abraham & Paprzycki, 2004). Watermarking is the process of embedding a specific copyright mark into digital documents in the same way. Nevertheless, in order to detect any break of licensing agreement, a serial number is embedded in every copy of this digital document. This process is known as fingerprinting.

Text steganography refers to the process of hiding secret information in text files. For security and imperceptibility reasons, it is very important for stego texts not to show any detectable artefacts. Thus, readers should not notice or discover the modifications made in the stego text files. Generally, the redundant information in text files is very limited in comparison to that in images and audio files. Therefore, using text as cover files in steganography represents the most difficult way of information hiding (Bender et al., 1996).

It is well known that the Web represents the world's premier network and Extensible Mark-up Language (XML) represents the world's premier data representation format (Newcomer, 2002). Though, Web services require a data exchange in the form of XML documents, Simple Object Access Protocol (SOAP) provides exactly this kind of data transport. Therefore, SOAP supports a common data transfer protocol for effective communication over the Web (Newcomer, 2002). Thus, XML is playing an increasingly important role in the exchange of a wide variety of data on the Internet. Therefore, XML documents are considered as a language of Web pages and digital contents. Moreover, they are used for the data exchange between organisations.

Basically, a SOAP message is an XML document that contains text. Therefore, steganography methods used for text files and XML documents can theoretically be used for SOAP messages. However practically, most of these methods are infeasible.

In this chapter, we propose a new steganography method to embed secret information in SOAP messages. This method changes the order of XML elements according to the secret message to be embedded.

The rest of this chapter is organised as follows. Section 6.3 reviews the related work on text and XML steganography. Section 6.4 discusses and explains the concept of information hiding within SOAP messages. Furthermore, our designed and proposed steganography method is illustrated in Section 6.5. An example scenario is illustrated in section 6.6. Finally, the conclusion is presented in Section 6.7.

6.3 Related Work

There is a relatively small number of text steganography studies in comparison to that of image video, and audio based steganography. This might be due to the lack of redundancy in text files (Inoue et al., 2001).

Basically, there are three major methods to hide data in text files. The first method, open space method, manipulates white spaces in the text. Therefore, it exploits inter-sentence spacing, end-of-line spaces, and inter-word spacing. The second method, syntactic method utilises punctuation. However, the third method, semantic method, manipulates the words of the text themselves (Bender et al., 1996).

Por & Delina (2008) improved the open space method proposed by Bender et al. (1996). Accordingly, they proposed a hybrid steganography method for text by combining both inter-word spacing and inter-paragraph spacing methods. Thus, whitespaces between words and paragraphs in right-justification of text are used for data hiding in order to increase the embedding capacity. However, the cover text was dynamically generated according to the size of the secret message.

Shirali-Shahrez (2008) proposed a new steganography method for texts. This method is based on the different spelling of some words in English between UK and US. For example, “centre” has different terms in UK (centre) and US (center). This can be used to hide a one bit each time a certain spelling occurs in the text. For example, A US spelling of a word means the secret bit is “0”, while a UK spelling means the secret bit is “1”.

Subsequently, the model proposed in (Shirali-Shahreza, 2008) defines a text steganography method based on substituting the words which have different terms in UK and US. For example, (Gas) has different terms in UK (Petrol) and US (Gas).

Liu, Guo & Zhou (2009) proposed a text steganography method to be used in online chat. This method is based on an Internet meme named typoglecymia, which holds that changing the order of word’s middle letters has a slight to no effect on the ability of skilled readers to understand the text (e.g. Guitar and Guiatr). Therefore, it used the redundancy found in the interior letters’ order. Since this letter randomisation equals to the common error made by chatters due to high speed typewriting, it is likely to be used in online chats, where the text usually contains mistakes.

However, there are fewer studies and examples of research regarding information hiding in XML files. Whilst the previous studies provide text steganography method, these are not necessarily applicable in SOAP messages context due to the fact that SOAP messages are exchanged and monitored by computer systems rather than humans. Importantly, using misspelled or alternative words in SOAP messages would result in the SOAP parsers not being able to handle the SOAP messages received because they do not comply with the expected semantic.

Inoue et al. (2001) proposed five steganography methods to be used with XML files. These steganography methods are summarised as follows:

1) The empty elements are represented according to the secret bit; either a start-tag immediately followed by an end-tag (<img></img>), or an empty-element tag (<img/>). This technique can embed one bit per empty element.

2) According to the secret bit, we can either add a white space before the close bracket (<tag >), or delete (normal with no added spaces) this white space (<tag>). This technique can embed one bit per tag.

3) Two elements may or may not be exchanged according to the secret bit. Thus, one bit per an exchange of two elements can be hidden.

4) The order of attributes in an element can be exchanged to hide the secret data. Thus, one bit per an exchange of the attributes order can be hidden.

5) Elements that contain each other can be used to hide secret data by exchanging inner-tags and outer-tags. In this method, one bit per an exchange can be hidden.

If an element has no content then empty-element tag can be used whether or not it is declared using the keyword EMPTY. However, the number of such elements in an XML document is limited and then the capacity of method (1) is limited too.

Additionally, using two formats to represent empty elements in the same document will arouse the attention of observers. Moreover, the parser may use only one representation of empty elements rather than two, which invalidate this method.

Whilst names of XML elements can't contain spaces, there can be a space before the closing character ">" (<tag >). However, this process will increase the size of the XML file and the hidden data may be destroyed due to parsing which may discard these added spaces (secret data).

Additionally, tags are case sensitive and therefore the tag <tag> is different from the tag <tag >. In other words, the end-tag’s name has to exactly match the start-tag's name. Thus, the method (2) is practically infeasible since it uses a start-tag different from the end-tag (one tag may contain a white-space).

The order in which attributes are included on an element is not considered relevant. For example, if an XML parser encounters a specific order of an element attributes, it does not necessarily have to give us the attributes in the same order. As a result, method (4) above is infeasible in terms of validity and applicability even though its capacity is very limited. However, a certain order of information can be maintained in an XML document if we put this information into elements, rather than attributes. As a result, method (3) above is a valid and possible solution for steganography. Nevertheless, hiding only one secret bit per an exchange of two elements represents a very small capacity.

Finally, an XML document must have a top-level element and all the other elements are its children. Furthermore, one and only one root element must be included in each XML document even if this element has no content. However, each of these children elements may represent a parent element and therefore have some sub-elements. Thus, exchanging a parent element with a sub-element technically looks valid (method (5) above). However, it seems impractical since the semantics will not make sense and we will get a new and different parent element by such an exchange. Also, the steganographic capacity of this method is very limited.

Since XML documents are widely used for data exchange over different networks and exposed to different threats, XML security become a key concern of organisations. Thus, Memon, Khawaja & Shah (2008) considered XML steganography as a new method and solution for secure communication. Furthermore, they proposed and designed four XML based steganography methods for the purpose of securing the cover file (XML document) rather than for the purpose of secret communication. The main aspects of these methods are as follows:

1) Random characters are inserted inside all tags and their values. So, after the first character of the first tag one random character is inserted, after the second character of the first tag two random characters are inserted and so on. Thus, it mixes up the actual XML data with random fake characters and therefore increases the size of the stego XML file significantly.

2) XML tags are shuffled (sequentially) in such a way the position of the 1st tag and its value are swapped with that of the last tag and its value. The same process happens with the second and the second last tags, and so on. The large XML file is, the better this technique work.

3) This is similar to the previous method, however the correct order of shuffled tags is identified in the attribute value of the root element. Thus, the first tag is determined by the first character of attribute value while the second character is randomly generated. Also, this method works better with large XML files.

4) The sequence of characters in all tags and values are reversed. Thus, the order of tags' characters is reversed by moving the last character to become the first one while the second last one becomes the second character and so on. As a result, the XML file will look like an encrypted file since the characters are scrambled in an unreadable form.

Memon, Khawaja & Shah (2008) then suggested combining all these methods together in one hybrid method to provide better XML security. In conclusion, all these four methods aim to safeguard the stego XML document against actual XML content detection rather than against hidden information detection. Additionally, their goal is the XML content not the hidden data itself. Therefore, the goal of these methods is totally different from our steganography goal which is undetectable and covert communication. Nevertheless, the first and fourth methods are definitely infeasible for steganography since the stego XML arouses the suspicion of everyone (look like encrypted). The second method may hide a few bits only, while in the third method, the secret key is included in the stego file, which is more than enough to extract the hidden message.

SOAP parsers have been developed and they only process XML that conforms to the SOAP schema and associated structural rules. Zhang, Wang & Sun (2007) proposed a steganography method depending on the text characteristics of SOAP technology in order to hide information in SOAP messages. Therefore, the physical properties of SOAP keywords and namespaces (self-defined) are used as cover message.

A character string is initialised by converting every letter in these keywords and namespaces into lowercases. Coordinating every secret bit with every letter of the character string, a specific letter is converted into a capital letter only when the secret bit is “1”. However, the amount of SOAP keywords is limited for short SOAP message.

Furthermore, the stego SOAP looks suspicious since some characters of this message are in lowercase shape while others are in uppercase shape. Therefore, the overall shape of the stego SOAP may attract attention. Last but not least, this method does not comply with the case-sensitivity nature of XML documents.

6.4 Information Hiding in SOAP Messages

The SOAP protocol is designed to enable the exchange of structured information (i.e. SOAP messages) over a variety of underlying protocols in decentralised and distributed environments. This lightweight protocol uses XML technologies to define a messaging framework that is independent of any specific programming languages or implementation semantics (Newcomer, 2002).

A SOAP message is an XML document, which consists mainly of “envelope, header, body and fault elements, as shown in (Figure 6.1). The “*Envelope*” is the root element that defines the XML document as a SOAP message. Also, it indicates the start and the end of the message. Application-specific information (such as security and reliability) is usually defined within the optional “*Header*” element. Additionally, headers may contain commands to SOAP processors either to understand these headers or to reject the SOAP message. However, the actual data is defined within the required “*Body*” element. Thus, mandatory information that must be delivered to the intended recipient should be included within the body part of SOAP message. The optional “*Fault*” element is used to identify error messages. If an error occurs during SOAP processing, a SOAP fault element will be emerge in the body of the message. Then, the sender of the SOAP message will get the fault response returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Header>…</S:Header>
   <S:Body>…
 <S:Fault>… </S: Fault >
   </S:Body>
</S:Envelope>
```

Figure 6.1: SOAP Message Construct

```
public class BookOrder{
   private String isbn;
   private String author;
   private String bookName;
   private int numOfPages;
   private String publisher;
   private int year;
   private double price;
  public getters and setters
}
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:BookOrder xmlns:ns2="http://service.bookorder.com/">
         <book>
<isbn>1-11-111111-1</isbn>
<author>Author_1</author>
<bookName>Book_1</ bookName >
<numOfPages>372</numOfPages>
<publisher>Publisher_1</publisher>
< year >2009</year>
<price>29.99</price>
         </book>
      </ns2:BookOrder>
   </S:Body>
</S:Envelope>
```

Figure 6.2: Example Java Class and Its XML Serialised Instance

When two parties communicate through SOAP messages, the actual data (i.e. fields and properties of objects or parameters and return values of methods) in the sender endpoint are converted (serialised) into an XML stream that conforms to the SOAP specifications. This serialised XML document is the SOAP message that needs to be de-serialised at the receiver endpoint to reconstruct the actual data. Figure 6.2 illustrates an example Java class *Book* and its XML serialised class instance.

An endpoint application normally employs a SOAP package to perform the serialisation and de-serialisation processes, as Web applications and clients care mainly about the actual data transmitted and not the structure of the SOAP message. Hence, secret information can be smuggled into SOAP messages, which provide a perfect cover if the hidden secret message does not damage the SOAP messages or spoil the actual data.

The main concern of hiding secret information within SOAP messages is how to do this without the fear of detection. Basically, end users care about the actual data transmitted but they do not care about other issues like SOAP namespace, keywords, or the order of elements' attributes. However, the transmitted message must generate no errors and therefore not to discard the message.

Hiding secret information in a SOAP message means that the mule that is used to convey the secret message is the communication protocol that governs the actual data path over a network, instead of using the actual data itself as a cover. This idea can overcome many of the limitations that faced the conventional steganography techniques. Traditional techniques hide secret messages inside digital files, which impose the threat of detecting the secret as these files are usually saved. Alternatively, a SOAP message leaves almost no trail as they are normally deleted after receiving the message and de-serializing the actual data. In addition, a secret piece of information can be divided into multiple smaller messages and transmitted over several SOAP messages to overcome the size limitation as well.

This chapter provides a novel steganography method that manipulates the SOAP protocol by rearranging the order of the contents and attributes of specific elements in a SOAP message, where every permutation represents a specific status according to a secret key shared between the sender and the receiver. For example, there are 7 sub-elements within the element *book* in Figure 6.2. These sub-elements are arranged in a particular order (*isbn, author, bookName, numOfPages, publisher, year, and price*). Whilst this order does not have any importance for the endpoint application, if the order of these sub-elements is rearranged, the message will still have the same meaning for the endpoint.

For a set of *n* sub-elements, there are a maximum of *n!* (factorial of *n*) permutations . This means that n! different sequences of order can be presented, and consequently, n! different symbols can be used in the character set of the stego script.

6.5 Steganography Framework for SOAP Messages

Considering the previous concept, we have designed and implemented a data hiding method that monitors a SOAP message just after its serialisation in the sender endpoint and before it is sent, analyses its elements and embeds a secret message accordingly. Figure 6.3 illustrates the general model of data hiding in SOAP messages.

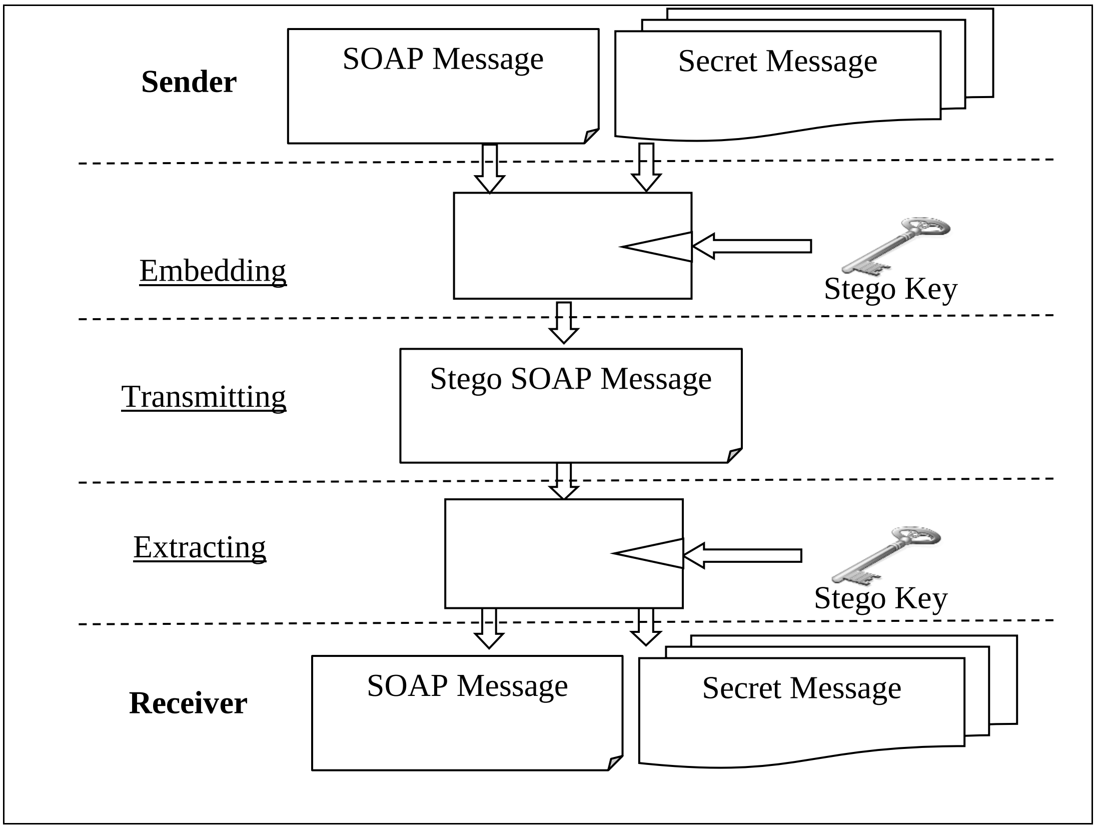

Figure 6.3: SOAP Steganography Model

When the stego SOAP message arrives at the receiver endpoint, the secret message is extracted using a stego key that is shared between the sender and receiver.

6.5.1 Embedding Procedures

In our proposed method, the procedure of hiding a secret message within SOAP consists of the following six steps.

1) Capturing the SOAP message after its serialisation.
2) Analysing its contents to identify all the elements with contents that can be rearranged to determine if the SOAP message is suitable for embedding (i.e. has elements with contents that can be rearranged).

3) Calculating the number of elements that can be used to hide data (N).

4) Permuting every set of sub-elements to reflect a status of a symbol from the secret message.

5) If all the symbols of the secret message can be hidden in one SOAP message (the number of available sets N is greater than the length of the secret message M), then the sub-elements of the set M+1 will be rearranged to indicate the end of secret message.

6) Otherwise, if M>N, only a part of the secret message is sent in this SOAP message and the last set of sub-elements is rearranged to indicate that more hidden data are to arrive within the next received SOAP message.

Figure 6.4 illustrates the algorithm used for secret message embedding.

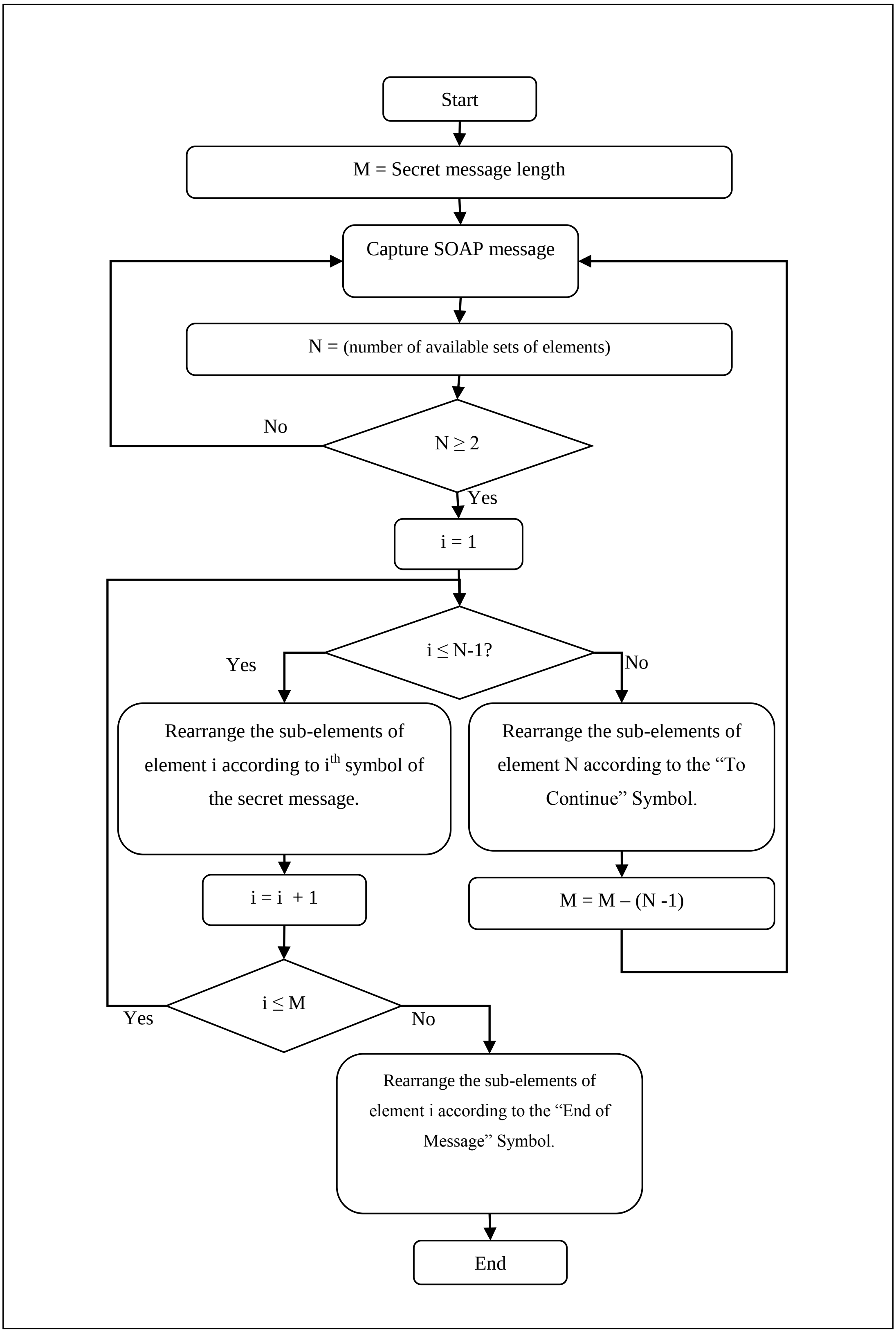
Start
M = Secret message length
Capture SOAP message
N = (number of available sets of elements)
N ≥ 2
No
Yes
i = 1
i ≤ N-1?
Yes
No
Rearrange the sub-elements of element i according to ith symbol of the secret message.
Rearrange the sub-elements of element N according to the "To Continue" Symbol.
i = i + 1
M = M – (N -1)
i ≤ M
Yes
No
Rearrange the sub-elements of element i according to the "End of Message" Symbol.
End

Figure 6.4: Secret Message Embedding

6.5.2 Extracting Procedure

The receiver, once the SOAP message is received, extracts hidden data by analysing the contents of each eligible element using the secret key to reveal the hidden symbol, as described in the following section and illustrated in the Extracting Algorithm (Figure 6.5):

1) Capturing the SOAP message and checking its validity and capability to be a stego SOAP message.

2) Calculating the number of elements that might be used for data hiding (N).

3) Extracting the hidden symbols by analysing the sub-elements order of each element in the stego SOAP message.

4) Stop the process either if the extracted symbol indicates that the message is not a stego SOAP or if the extracted symbol means “End of Message”.

5) If the extracted symbol means “To Continue”, new SOAP message to be captured and analysed as in 1.

6) Otherwise, the next symbol will be extracted and so on until we get the entire secret message embedded.

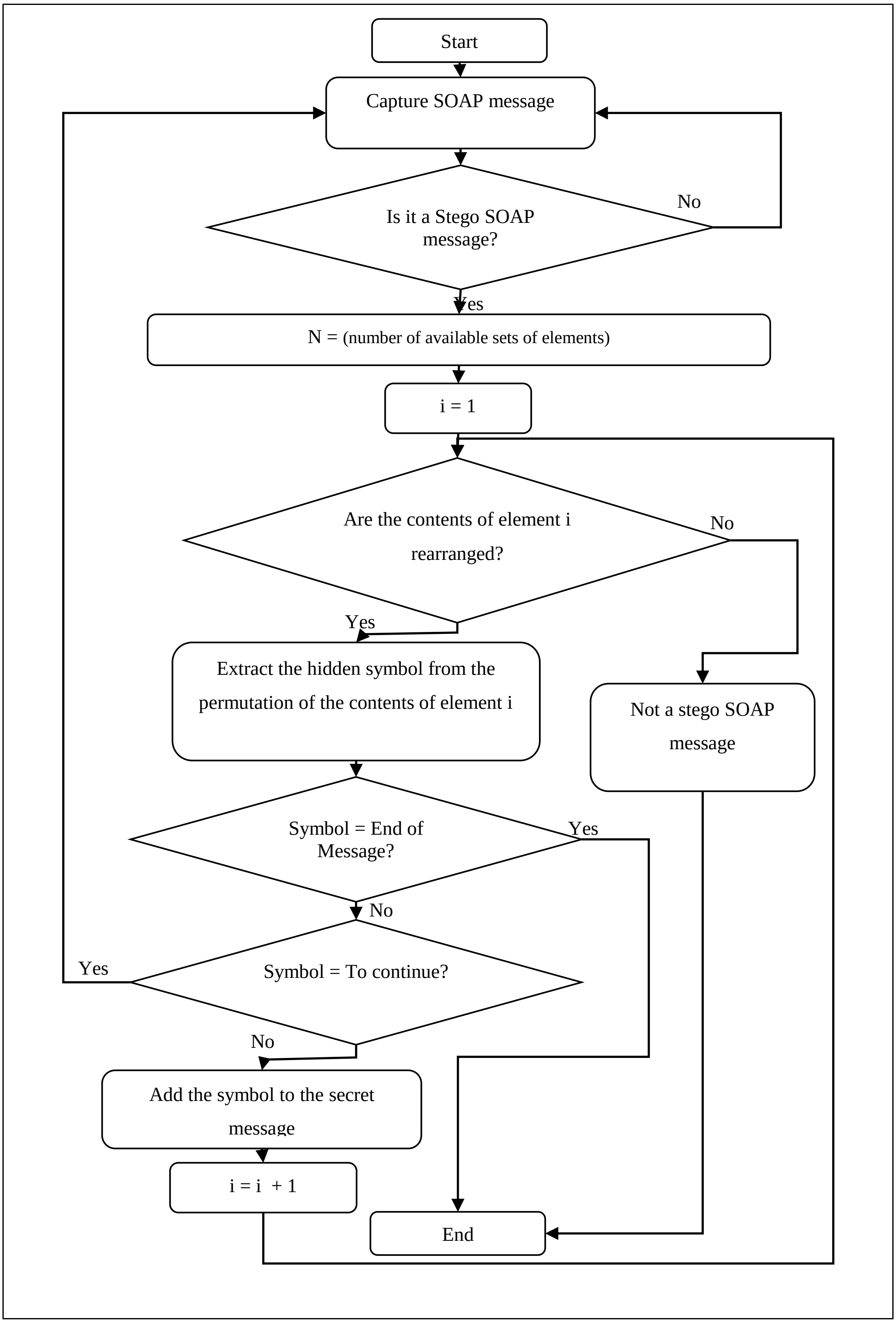
Start
Capture SOAP message
Is it a Stego SOAP message?
No
Yes
N = (number of available sets of elements)
i = 1
Are the contents of element i rearranged?
No
Yes
Extract the hidden symbol from the permutation of the contents of element i
Not a stego SOAP message
Symbol = End of Message?
Yes
No
Yes
Symbol = To continue?
No
Add the symbol to the secret message
i = i + 1
End

Figure 6.5: Secret Message Extracting

6.6 Example Scenario

Our proposed method for SOAP message-based steganography is empirically tested and validated. Thus, we demonstrate the data embedding and extracting algorithms using an example, yet realistic, Web service scenario (Book Order): In this scenario, we assume that the person who wants to send secret data is the "Book Buyer" while the intended recipient of secret message is the "Book Seller". However, the opposite scenario is true since the "Book Seller" can send a secret message to the "Book Buyer" using the same procedure. The example scenario is:

Step 1: The Book Buyer (Service Requester) selects the books to be ordered from the Book Seller Website (service Provider).

Step2: The Book Order will be formatted as XML document and then an XML-based SOAP message will be generated in order to be sent to the Service Provider.

Step 3: An application is used at the sender (Book Buyer) endpoint in order to capture each SOAP message before it has been sent (prevents the sending process of SOAP).

Step 4: The "Embedding Procedure" of our SOAP steganography method is applied on each captured SOAP message.

Step 5: The output of the "Embedding procedure" (a stego SOAP message) will be sent to the Book Seller.

Step 6: The Book Seller receives the SOAP message (a stego SOAP) and a similar application to that used at the Book Buyer endpoint will be used at the Book Seller endpoint to capture each received SOAP message.

Step 7: The "Extracting Procedure" of our SOAP steganography method is applied on each captured SOAP message in order to extract the secret message from the stego SOAP messages.

As illustrated in Figure 6.6, a book buyer is sending two messages to the book seller. The first SOAP message contains an order for four books, while the second is an order for three books. A secret message "*Hello*" is smuggled by shuffling the sub-elements of each "*book*" element in these SOAP messages. The first message contains only part of the hidden message "*Hel*" and "*to continue*" symbol (Figure 6.7), while the second message contains the rest of the message *"lo"* and the "*end of message*" symbol (Figure 6.8). Because each element has five sub-elements, 5! (=120) different cases can be represented. That covers all the alphabetical characters (in small and capital caps), numbers and most of the printing characters. For the purpose of demonstration, we used a shifted version of the ASCII table as a secret key for data hiding, see Appendix D. More complex secret keys can be used in real implementations.

For this experiment, NetBeans IDE 6.9 is used to develop the Web service (book seller service) and the client (book buyer application). The Web service is built as a Web application and deployed on a Glass Fish 2.2 application server. All the SOAP messages are intercepted in the sender endpoint just after being serialised into XML messages and before the SOAP messages are sent to the receiver endpoint. Similarly, all the coming SOAP messages are intercepted before they are de-serialised. The SOAP messages are also monitored and recorded using soapUI in the standard HTTP proxy mode.

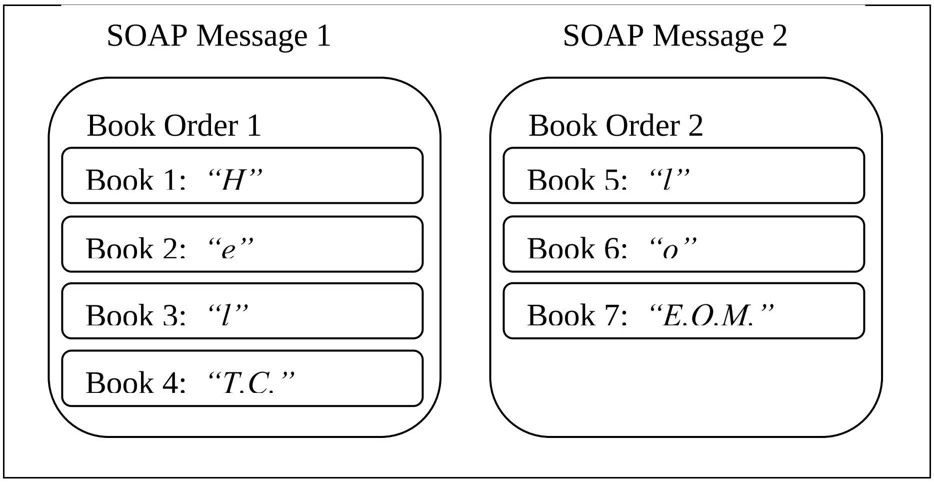

Figure 6.6: Example Secret Message Hidden in Two SOAP Messages

Book Order (Step 1):
Order 1:
1-Book 1: isbn=1-11-111111-1, author=Author_1, name=Book_1, pages=111, publisher=Publisher_1
2-Book 2: isbn=2-22-222222-2, author=Author_2, name=Book_2, pages=222, publisher=Publisher_2
3-Book 3: isbn=3-33-333333-3, author=Author_3, name=Book_3, pages=333, publisher=Publisher_3
4-Book 4: isbn=4-44-444444-4, author=Author_4, name=Book_4, pages =444, publisher=Publisher_4

Cover SOAP (Stes 2+3):

SOAP Message 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns2:bookOrder
xmlns:ns2="http://service.testproject/">
   <book>
     <isbn>1-11-111111-1</isbn>
     <author>Author_1</author>
     <name>Book_1</name>
     <pages>111</pages>
     <publisher>Publisher_1</publisher>
   </book>
   <book>
     <isbn>2-22-222222-2</isbn>
     <author>Author_2</author>
     <name>Book_2</name>
     <pages>222</pages>
     <publisher>Publisher_2</publisher>
   </book>
   <book>
     <isbn>3-33-333333-3</isbn>
     <author>Author_3</author>
     <name>Book_3</name>
     <pages>333</pages>
     <publisher>Publisher_3</publisher>
   </book>
   <book>
     <isbn>4-44-444444-4</isbn>
     <author>Author_4</author>
     <name>Book_4</name>
     <pages>444</pages>
     <publisher>Publisher_4</publisher>
   </book>
  </ns2:bookOrder>
 </S:Body>
</S:Envelope>
```

Stego SOAP (Steps 4+5+6+7):

SOAP Message 1:

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns2:bookOrder
xmlns:ns2="http://service.testproject/">
   <book>
     <author>Author_1</author>
     <pages>111</pages>
     <publisher>Publisher_1</publisher>
     <name>Book_1</name>
     <isbn>1-11-111111-1</isbn>
   </book>
   <book>
     <name>Book_2</name>
     <publisher>Publisher_2</publisher>
     <pages>222</pages>
     <isbn>2-22-222222-2</isbn>
     <author>Author_2</author>
   </book>
   <book>
     <pages>333</pages>
     <isbn>3-33-333333-3</isbn>
     <publisher>Publisher_3</publisher>
     <name>Book_3</name>
     <author>Author_3</author>
   </book>
   <book>
     <publisher>Publisher_4</publisher>
     <isbn>4-44-444444-4</isbn>
     <author>Author_4</author>
     <name>Book_4</name>
     <pages>444</pages>
   </book>
  </ns2:bookOrder>
 </S:Body>
</S:Envelope>
```

Secret Message:
“Hel”
Stego Key:
NO EMBEDDING = isbn, author, name, pages, publisher.
“H” Character = author, pages, publisher, name, isbn
“e” Character = name, publisher, pages, isbn, author
“l” Character = pages, isbn, publisher, name, author
“To Continue” = publisher, isbn, author, name, pages

Figure 6.7: Hiding the First Part of the Secret Message

Book Order (Step 1):
Order 2:
1-Book 5: isbn=5-55-555555-5, author=Author_5, name=Book_5, pages =555, publisher=Publisher_5
2-Book 6: isbn=6-66-666666-6, author=Author_6, name=Book_6, pages =666, publisher=Publisher_6
3-Book 7: isbn=7-77-777777-7, author=Author_7, name=Book_7, pages =777, publisher=Publisher_7

Cover SOAP (Stes 2+3):

```
SOAP Message 2:
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
<ns2:bookOrder
xmlns:ns2="http://service.testproject/">
         <book>
            <isbn>5-55-555555-5</isbn>
            <author>Author_5</author>
            <name>Book_5</name>
            <pages>555</pages>
            <publisher>Publisger_5</publisher>
         </book>
         <book>
            <isbn>6-66-666666-6</isbn>
            <author>Author_6</author>
            <name>Book_6</name>
            <pages>666</pages>
            <publisher>Publisger_6</publisher>
         </book>
         <book>
            <isbn>7-77-777777-7</isbn>
            <author>Author_7</author>
            <name>Book_7</name >
            <pages>777</pages>
            <publisher>Publisher_7</publisher>
         </book>
      </ns2:bookOrder>
   </S:Body>
</S:Envelope>
```

Stego SOAP (Steps 4+5+6+7):

```
SOAP Message 2:
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
<ns2:bookOrder
xmlns:ns2="http://service.testproject/">
         <book>
            <pages>555</pages>
            <isbn>5-55-555555-5</isbn>
            <publisher>Publisger_5</publisher>
            <name>Book_5</name>
            <author>Author_5</author>
         </book>
         <book>
            <pages>666</pages>
            <author>Author_6</author>
            <name>Book_6</name>
            <isbn>6-66-666666-6</isbn>
            <publisher>Publisger_6</publisher>
         </book>
         <book>
            <publisher>Publisher_7</publisher>
            <pages>777</pages>
            <name>Book_7</name >
            <author>Author_7</author>
            <isbn>7-77-777777-7</isbn>
         </book>
      </ns2:bookOrder>
   </S:Body>
</S:Envelope>
```

Secret Message:
“lo"
Stego Key:
NO EMBEDDING = isbn, author, name, pages, publisher.
“l” Character = pages, isbn, publisher, name, author
“o” Character = pages, author, name, isbn, publisher
“End of Message” = publisher, pages, name, author, isbn

Figure 6.8: Hiding the Second Part of the Secret Message

6.7 Summary

In this chapter, we have provided a communication protocol-based steganography method that manipulates the SOAP protocol. This method monitors a SOAP message just after its serialisation in the sender endpoint and before it is sent. It analyses the SOAP elements and embeds a secret message accordingly by rearranging the order of the contents and attributes of specific elements in a SOAP message, where every permutation represents a specific symbol according to a secret key shared between the sender and the receiver. As a result, the provided method has a high resistance against detection since it uses the communication protocol as a cover medium rather than the traditional digital files. Furthermore, the stego SOAP message has the same size of the original message. The method is tested and validated using a feasible scenario so as to demonstrate its utility and applicability.

Security is an ongoing process and as soon as developers fix one set of problems crackers will find yet another way to break these systems. Essentially, the applications must be flexible in order to add new security features as needed. Furthermore, anyone on the Internet can intercept the data transmitted between different sites. Thus, distributed applications require higher security levels than internal applications.

Encryption can be used to preserve data security but the technologies required for encryption cause problems with firewalls and they don't work very well on the Internet. Encryption has another problem; if both communication parties don't have the same platform then the receiver can't decrypt the sender's message. Thus, even a common encryption scheme usually can only work on a limited number of platforms (Mueller, 2001). As a result, our SOAP based steganography method could be a reasonable solution for transmitted data security. It can be used as a secret communication channel over different kinds of networks regardless of the applications used at the distributed endpoints. As a kind of communication security, the process of surely knowing the identity of the other communicating party (on the other end of a channel) is known as Authentication. Additionally, associated HTTP Authentication Framework with HTTP 1.1 provides better authentication means between communicating parties. Thus, the HTTP Authentication Framework secures only the authentication portion of the communication. Furthermore, Secure/Multipurpose Internet Mail Extensions (S/MIME) and Secure Socket Layer (SSL) use digital certificates to provide security which relies on the use of public key cryptography. Usually, using static keys provides the crackers more chance to break the system than using dynamic keys (Mueller, 2001).

As a result, we can use the proposed SOAP steganography method to convey information of authentication which necessary to authenticate the communicating parties. Additionally, encryption keys can be embedded and transmitted in order to get dynamic keys instead of static keys, and therefore add another level of system security.

On the other hand, the proposed method does not specify any prior stego keys as it assumes that the initial exchange of the original stego key occurs *out-of-band*. Hence, traditional cryptography-based key management mechanisms can be used to facilitate the sharing of the stego key, and the overhead of the key sharing relies mainly on the selected key management approach. For example, a simple username authentication mechanism can be used to generate a symmetric stego key that is used for both embedding and extracting hidden messages within the SOAP messages in all future communications between the client and the server. For this approach, the client does not possess any key of its own, but instead authenticate to the server using a username/password token, which is used to generate a single shared stego key at run time. Alternatively, more advanced STS-based approaches may be used to generate and share a sequence of stego keys, where each key is valid for a certain period of time or number of exchanged messages.

Basically, encryption algorithms represent a conventional solution of information security but the encrypted data is still there and everyone can observe it over the network. Thus, our SOAP steganography algorithm provides a way of secret communications over the Internet. It can overcome the limitations and challenges of encryption as well as it can be used with encryption to provide a double layer of security.

In conclusion, the proposed SOAP steganography method can be used for a variety of applications such as; authentication, proof of identity, watermarking, and message hashing. The framework is the outcome of the fourth iteration of the DSR cycle (see section 3.6.4)

Chapter 7: Conclusion

7.1 Overview

Web services technology is the modern way of connecting businesses. Companies nowadays expose their business functionalities, via the Web, as services for their customers or other companies to use. However, this exposure represents a complex dilemma of finding the right balance between security and performance. This is because developers have to secure these Web services to limit their vulnerability to attacks, while ensuring that an acceptable level of performance is provided by these services. In view of that, the aim of the research presented in this thesis was to develop a prototypical framework that aids Web services architects to select the best suited security approach that satisfies the security and performance requirements of a given Web services application.

In this chapter, we summarise the research conclusions and findings and identify the research limitations in order to provide future research directions.

7.2 Thesis Overview and Findings

This thesis was organised in seven chapters. The following section summarises the previous first six chapters:

Chapter 1 is the introduction of this thesis, in which the main motivations for conducting this research were explored. The discussion highlighted the impact of applying security on the performance of Web services and indicated that whilst the different Web services security profiles achieve different levels of security, they also have different performance measurements. This emphasised the importance of selecting a suitable security approach that provides the best balance between security and performance, in accordance with the system requirements and limitations. Consequently, the aim of this research was identified as developing a framework to help Web services architect to choose the best suited security profile that fulfils the security and performance expectations of a given Web services application. Thereafter, the steps to achieve this aim were identified as the objectives of this thesis and research approach was briefly explained.

Chapter 2 provided an overview of Web services security and its related standards. Several approaches to analyse the performance of Web services security were discussed and the trade-off between security and performance was identified as a major challenge that faces the development of secure Web services. The literature review indicates that this trade-off varies depending on the applied security approach. Furthermore, there are many cases that require employing a combination of security specifications to achieve the required level and coverage of security. Web services development environments normally provide developers with predefined sets of security profiles and allow for their parameters to be adjusted according to the security preferences of the developed system. The chapter pointed towards the importance of evaluating the security coverage of these profiles against their performance when selecting a security profile for a Web service, which requires a performance testing model as well as a systematic framework to rate these security profiles.

Chapter 3 explained the research method undertaken in this thesis. A theoretical grounding of Design Science Research (DSR) is provided in this chapter. Thereafter, The DSR paradigm is justified as a suitable approach for this research. The research conducted in this thesis is then explained in line with the DSR research cycle. Four iterations were identified and presented to accomplish the development of the selection model: (1) *Library Research*, (2) *Laboratory Experiments*, (3) *Configuration Requirements* and (4) *Alternative Solution Based on Steganography*.

Chapter 4 compared the performance of several security profiles for Web services. The performance evaluation indicated that profiles that use transport level security are always faster than message level security profiles. In addition, Message level security protocols have a scalability problem when large messages are exchanged, unlike SSL-based profiles. Security profiles that utilise username tokens perform better than mutual certificates security, especially when exchanging small size messages. Moreover, the performance of SAML-based profiles is controlled by their underlying security profile. STS-based security profiles perform massively less than non-STS profiles and should only be considered when the service and its client are located in different domains. Finally, reliability has a huge impact on the performance of Web services as it increases the number and size of SOAP messages, due to the addition of reliability assertions, as well as the process time.

Chapter 5 proposed a multi-criteria decision making framework, based on the Analytical Hierarchy Process (AHP) approach. This framework incorporates not only the security requirements, but also the performance considerations as well as the configuration constraints of these security profiles. This approach emphasises the service developer viewpoint and focuses on analysing the performance of the security profiles rather than the individual security features. The framework is then empirically validated and evaluated using a scenario-driven approach to demonstrate its utility and value. The different scenarios illustrate various situations where the decision making framework is used to make informed decisions to rank various security profiles in order to select the most suitable one for each scenario.

Chapter 6 provided an alternative approach to secure Web services based on data hiding (steganography). This approach utilises a communication protocol-based steganography method that manipulates the SOAP protocol by analysing the SOAP message and hiding a secret message accordingly. The idea is based on rearranging the order of the XML tags of specific elements in a SOAP message. Every permutation of these tags represents a specific symbol according to a secret key shared between the sender and the receiver. This steganographical approach has a high resistance against detection since it uses the communication protocol as a cover medium rather than the traditional digital files and keeps the original size of the SOAP message intact. This method is then illustrated using a feasible scenario to demonstrate its applicability and effectiveness.

7.3 Research Contribution

This thesis contributes to the theory and the demonstration of theory in practice. The integration of the diverse but interconnected domains of Web services, security, performance analysis, decision making and steganography enriches the quality of this research. The main contributions of this thesis are as follows:

7.3.1 A Performance Evaluation Model

This model builds upon current research and extends it to provide guidelines for Web services developers to aid them when selecting a security approach for their applications. The proposed model helps to establish a better understanding of the trade-off between security and performance in the field of Web services research by measuring the performance of the security profiles and comparing the results to classify the various security profiles according to their performance.

Several studies tried to tackle the problem of mapping security to performance in the area of Web services (Shirasuna et al., 2004; Moralis et al., 2007; Novakouski et al, 2010), but they generally compared the performance of different specifications and standards that are rarely used in a stand-alone manner. Instead, security profiles are normally used as they combine standards that are guaranteed to work in a harmony. Therefore, the approach we followed to develop this model was to evaluate the performance of these security profiles, rather than the underlying standards. This approach shields the developers from the complexity of these standards and allows them to direct their efforts towards fulfilling the security as well as performance requirements of their applications.

In addition, our proposed model is more comprehensive than those developed in previous research as we tested a larger number of security and reliability profiles that represent various security methods using different message sizes.

7.3.2 A Multi-Criteria Decision Making Framework

The second contribution have provided a novel multi-criteria decision making framework, and accompanying software, based on the analytical hierarchy process in order to help developers to rate security profiles according to the security, performance and configuration requirements. The approach we followed to develop this framework is to focus on the service developer viewpoint during the development process. This is different from other related work that studied Web services composition (Wu & Chang, 2007; Zuo, Wang & Wu, 2008; Godse, Sonar & Mulik, 2008; Casola et al., 2009; Thirumaran et al.; 2011), and therefore focused on the service consumer point of view. This framework provides developers with a useful tool for the selection of security profiles for Web services applications. This particularly useful in new or unusual cases, where no or very little documentations of best practice are available. There are cases where several profiles can be implemented, or when all the profiles cannot entirely satisfies all the requirements, where this tool can help to determine the best option.

We have also provided three common security usage scenarios for Web services and tested them using the proposed framework to demonstrate its validity and effectiveness. The framework results match the security recommendations for these scenarios.

7.3.3 A Steganographic Method for Web Services

In this thesis, we provide an alternative approach to secure Web services based on a novel Steganography method. This method was developed to overcome the limitations of using cryptographic-based techniques. The first major issue we identified with the traditional security and reliability approaches is their impact on the size of SOAP-messages, as illustrated in Appendix A. In addition although encryption can be used to establish data security but the technologies required for encryption are not always firewall-friendly, and they are in many cases platform-dependant (Mueller, 2001).

Therefore, we provided a communication protocol-based steganography method that manipulates the SOAP protocol by monitoring a SOAP message just after its serialisation in the sender endpoint and before it is sent, analysing the SOAP elements and embedding a secret message accordingly. This is done by rearranging the order of specific elements in the SOAP message in a way where every permutation represents a specific character according to a secret key shared between the sender and the receiver. This method has a high resistance against detection because it uses the communication protocol as a cover medium rather than the traditional digital files. The proposed data hiding method produces stego SOAP messages that have exactly the same size of the original message, which makes it undetectable using conventional detecting methods. Furthermore, a secret message can be sent over a number of messages, providing that there is a continues interaction between the sender and the receiver, which overcomes the capacity issue of traditional data hiding techniques that are limited by the size of the cover medium. There are several applications that can benefit from this method, such as authentication and watermarking.

In conclusion, this research has suggested a prototypical framework for selecting a security profiles for Web services applications. This framework consists of three approaches: (1) a performance analysis approach that provides Web services developers with a performance guideline, (2) a multi-criteria decision making approach that aids developer to make an informed decision when selecting a security profile based on different requirements and limitations, and finally (3) a steganographic method that can be used as an alternative to cryptographic-based security methods.

Table 7-1 shows how this research successfully accomplished the objectives established in section 1.3.

Research Objective	Accomplishments
Objective 1: Design a performance testing model to understand the cost of applying security profiles on the performance of Web services.	The first objective was achieved in Chapter 4: Several experiments were conducted to compare the performance of security profiles. The results were analysed and presented.
Objective 2: Develop a decision making framework based on the results gathered from the testing model as well as the security requirements and system limitations.	We accomplished these objectives in Chapter 5: A MCDM framework based on the AHP was developed. A generic AHP software tool was also developed to be used as an instantiation of the framework. Three different scenarios were used to validate the framework.
Objective 3: Evaluate the developed framework through the use of real scenarios so as to indicate the framework's utility and value.	
Objective 4: Explore and evaluate the feasibility of using steganography as an alternative approach to secure Web services.	This objective was met in Chapter 6: We have developed an algorithm for embedding and extracting hidden messages in/from SOAP messages using a novel steganographic method. This method is based on XML-tags shuffling. The method is validated through the use of a real scenario.

Table 7-1: Accomplishments of The Research Objectives

7.4 Research Limitations and Future Work

During this course of this study, few issues were identified as limitations that may require further addressing in future work.

The first limitation of this research is the number of security profiles we evaluated, although we carefully selected a number of security profiles so they are representatives of the different security methods, token types, security layer and security coverage. We consider that there are more profiles that can be tested to enhance the performance testing model. These profiles can also be tested using different configurations, platforms and implementations so as to expand the scope of this model.

The AHP framework can be improved by conducting rigorous pair-wise comparisons using a number of experts in the field of Web services security, and then integrating the group judgment in one overall framework. It is also advantageous to conduct further validations and consider expanding the AHP hierarchy by including other criteria and sub-criteria, such as cost and usability.

This thesis has also presented a data hiding method based on XML shuffling technique. However, this method can only be applied to SOAP messages that have a complex payload structure (in order to have enough XML tags to shuffle). Messages with simple payload structures cannot be used with this method in its current form. Exploring other approaches to hide data in Web services may present an interesting future research. In addition, this method can be further developed and implemented to be used in a variety of applications, such as watermarking, establishing identity and reliable messaging.

References

Adams, C. & Boeyen, S. (2002) 'UDDI and WSDL Extensions for Web Service: A Security Framework', *Proceedings of the 2002 ACM workshop on XML security, Fairfax, Virginia, USA, 22 November 2002*, pp. 30-35.

Albreshne, A., Fuhrer, P. & Pasquier, J. (2009) *Web Services Technologies: State of the Art*, University of Fribourg, Switzerland. Available at: http://diuf.unifr.ch/drupal/softeng/sites/diuf.unifr.ch.drupal.softeng/files/file/publications/internal/WP09-04.pdf.

Al-Debei, M.M. & Fitzgerald, G. (2010) 'The Design and Engineering of Mobile Data Services: Developing an Ontology based on Business Model Thinking' in *Human Benefits Through the Diffusion of Information Systems Design Science Research*, eds. J. Pries-Heje, J. Venable & J. De Gross. Boston: Springer, pp. 28-51.

Ashayeri, J., Keij, R. & Bröker, A. (1998) 'Global business process re-engineering: a system dynamics-based approach', *International Journal of Operations & Production Management*, 18 (9/10), pp. 817-831.

Badria, M.A. & Davisb, D. (2001) 'A comprehensive 0-1 goal programming model for project selection', *International Journal of Project Management*, 19 (4), pp. 243-252.

Bajaj, S., Box, D., Chappell, D., Curbera, F., Daniels, G., Hallam-Baker, P., Hondo, M., Kaler, C., Langworthy, D., Nadalin, A., Nagaratnam, N., Prafullchandra, H., von Riegen, C., Roth, D., Schlimmer, J., Sharp, C., Shewchuk, J., Vedamuthu, A., Yalçinalp, Ü. & Orchard, D. (2006) *Web Services Policy 1.2 - Framework (WS-Policy)*, W3C. Available at:http://www.w3.org/Submission/WS-Policy/.

Baker, D., Bridges, D., Hunter, R., Johnson, G., Krupa, J., Murphy, J. & Sorenson, K. (2001) *Guidebook to decision-making methods*, Washington DC, USA: Department of Energy.

Bender, W., Gruhl, D., Morimoto, N. & Lu, A. (1996) 'Techniques for Data Hiding', *IBM Systems Journal*, 35 (3-4), pp. 313-336.

Bertino, E., Martino, L., Paci, F. & Squicciarini, A. (2010) *Security for Web Services and Service-Oriented Architectures*, Berlin: Springer.

Bhushan, N. & Rai, K. (2004) *Strategic decision making*, Springer.

Booch, G. (1986) '*Object-Oriented Development*', *IEEE Transactions on Software Engineering*, 12 (2), pp. 211-221.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. & Orchard, D. (2004) *Web Services Architecture*, W3C.

Brown, A., Johnston, S. & Kelly, K. (2002) *Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications*, Rational Software Corporation.

Buss, M.D.J. (1983) 'How to rank computer projects', *Harvard business review*, 61 (1), pp. 118-125.

Carminati, B., Ferrari, E. & Hung, P.C.K. (2005) 'Exploring Privacy Issues in Web Services Discovery Agencies', *IEEE Security & Privacy*, 3 (5), pp. 14-21.

Carr, H. & Guo, J. (2009) *Metro Web Services Security Usage Scenarios*, Sun Microsystems. Available at: http://jazoon.com/portals/0/Content/ArchivWebsite/jazoon.com/jazoon09/download/presentations/9021.pdf.

Casola, V., Fasolino, A.R., Mazzocca, N. & Tramontana, P. (2009) 'An AHP-Based Framework for Quality and Security Evaluation', *Proceedings of the International Conference on Computational Science and Engineering, CSE '09*, pp. 405-411.

Cerami, E. (2002) *Web Services Essential*, 1st ed. edn, Sebastopol, CA: O'Reilly & Associates, Inc.

Chen, S., Zic, J., Tang, K. & Levy, D. (2007) 'Performance Evaluation and Modeling of Web Services Security', *Proceedings of the IEEE International Conference on Web Services, ICWS 2007*, pp. 431-438.

Christensen, E., Curbera, F., Meredith, G. & Weerawarana, S. (2001) *Web Services Description Language (WSDL) 1.1*, W3C. Available at: http://www.w3.org/TR/wsdl.

Chua, W.F. (1986) 'Radical Developments in Accounting Thought', *The Accounting Review*, 61 (4), pp. 601-632.

Clement, L., Hately, A., von Riegen, C. & Rogers, T. (2004) *UDDI Spec Technical Committee Draft*, OASIS. Available at: http://uddi.org/pubs/uddi_v3.htm.

Dodds, L. (2002) 16 Jan. 2002-last update, *Fat Protocols.* O'Reilly. Available at: http://www.xml.com/pub/a/2002/01/16/deviant.html.

Dolan, J.G. (1989) 'Medical Decision Making Using the Analytic Hierarchy Process: Choice of Initial Antimicrobial Therapy for Acute Pyelonephritis', *Med Decis Making*, 9 (1), pp. 51-56.

Dutta, R. & Burgess, T.F. (2003) 'Prioritising information system projects in higher education', *Campus-Wide information system*, 20 (4), pp. 152-158.

Edwards, W. & Barron, F.H. (1994) 'Smarts and smarter: Improved simple methods for multiattribute utility measurement', *Organizational Behavior and Human Decision Processes*, 60 (3), pp. 306-325.

Geer, D. (2003) 'Taking Steps to Secure Web Services', *Computer*, 36 (10), pp. 14-16.

Ghinea, G., Magoulas, G.D. & Siamitros, C. (2005) 'Multicriteria decision making for enhanced perception-based multimedia communication', *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35 (6), pp. 855-866.

Godse, M., Sonar, R. & Mulik, S. (2008) 'The Analytical Hierarchy Process Approach for Prioritizing Features in the Selection of Web Service', *Proceedings of the IEEE Sixth European Conference on Web Services, ECOWS '08.* , pp. 41-50.

Goncalves, A. (2010) *Beginning Java EE 6 Platform with GlassFish 3: From Novice to Professional*, 2nd Edition edn, USA: Apress.

Goodwin, P. & Wright, G. (1999) *Decision analysis for management judgment*, Wiley.

Gray, N.A.B. (2004) 'Comparison of Web Services, Java-RMI, and CORBA service implementations', *Proceedings of the Fifth Australasian Workshop on Software and System Architectures*.

Guba, E.G. & Lincoln, Y.S. (1994) 'Competing paradigms in qualitative research'. In Handbook of qualitative research' in *Handbook of qualitative research*, eds. N.Y.K. Denzin & Y.S. Lincoln,Thousand Oaks: Sage publications, pp. 105-117.

Guitart, J., Beltran, V., Carrera, D., Torres, J. & Ayguade, E. (2005) 'Characterizing Secure Dynamic Web Applications Scalability', *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, Denver, CO, USA: IEEE, pp. 108-121.

Guruge, A. (2003) *Web Services, Theory and Practice*, Oxford: Elsevier Inc.

Hatala, M., Eap, T. & Shah, A. (2004) 'Secure Communication Infrastructure for Object Repositories and Web Services', *Proceedings of the 2nd Annual Lornet Conference, Vancouver, BC, 15 November 2005*.

He, H. (2003) 'What is Service-Oriented Architecture?', pp. 15 December 2005. Available at: http://www.xml.com/pub/a/ws/2003/09/30/soa.html.

Head, M.R., Govindaraju, M., Slominski, A., Pu Liu, Abu-Ghazaleh, N., van Engelen, R., Chiu, K. & Lewis, M.J. (2005) 'A Benchmark Suite for SOAP-based Communication in Grid Web Services', *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, pp. 19-19.

Head, M.R., Govindaraju, M., van Engelen, R. & Zhang, W. (2006) 'Benchmarking XML processors for applications in grid web services', *Proceedings of the 2006 ACM/IEEE conference on Supercomputing, Tampa, Florida*, New York, NY, USA: ACM.

Hevner, A.R., March, S.T., Park, J. & Ram, S. (2004) 'Design Science in Information Systems Research', *Management Information Systems Quarterly*, 28 (1), pp. 75-105.

Holgersson, J. & Soderstrom, E. (2005) 'Web service security - vulnerabilities and threats within the context of WS-security', *The 4th Conference on Standardization and Innovation in Information Technology (SIIT2005), 2005.* Geneva, Switzerland, pp. 138-146.

Hondo, M., Nagaratnam, N. & Nadalin, A. (2002) 'Securing Web Services', *IBM Systems Journal*, 41 (2), pp. 228-241.

Iivari, J. (2007) 'A Pragmatic Analysis of Information Systems as a Design Science', *Scandinavian Journal of Information Systems*, 19 (2), pp. 39-64.

Inoue, S., Makino, K., Murase, I., Takizawa, O., Matsumoto, T. & Nakagawa, H. (2001) 'A proposal on information hiding methods using XML', *Proceedings of the 1st Workshop of NLP and XML, Nov, 2001*, pp. 43.

Jeckle, M., Melzer, I. & Himsolt, M. (2004) 21 Feb. 2004-last update, *Performance of Web Services.* Faculty of Mathematics and Economics, ULM University. Available at: http://www.mathematik.uni-ulm.de/sai/ws03/webserv/PerfWS.pdf.

Jensen, R.E. (1982) 'Reporting of management forecasts: An eigenvector model for elicitation and review of forecasts', *Decision Sciences*, 13 (1), pp. 15-37.

Jensen, R.E. & Spencer, R.W. (1986) 'Matrix scaling of subjective probabilities of economic forecasts', *Economics Letters*, 20 (3), pp. 221-225.

Kamal, M.M. (2008) 'Investigation Enterprise Applications Integration (EAI) Adoption in the Local Goverment Authorities (LGAs)'.

Karsak, E.E., Sozer, S. & Alptekin, S.E. (2003) 'Product planning in quality function deployment using a combined analytic network process and goal programming approach', *Computers & Industrial Engineering*, 44 (1), pp. 171-190.

Kepner, C.H. & Tregoe, B.B. (1981) *The new rational manager*, Princeton Research Press.

Kim, J. & Mueller, C.W. (1993) *Factor analysis: statistical methods and practical issues,* 19th edn, Sage Publ.

Klein, K.K. & Myers, M.D. (1999) 'A Set of Principles for Conducting and evaluating Interpretive Field Studies in Information Systems', *MIS Quarterly*, 23 (1), pp. 67-94.

Kontogiannis, K. (2008) 'Challenges and opportunities related to the design, deployment and, operation of Web Services', *Proceedings of the 2008 Frontiers of Software Maintenance (FoSM 2008)*, Beijing, China, pp. 11-20.

Kuechler, B. & Vaishnavi, V. (2008) 'On Theory Development in Design Science Research: Anatomy of a Research Project', *European Journal of Information Systems*, 17 (5), pp. 489-504.

Kuhn, T. (1996) *The Structure of Scientific Revolutions*, Chicago: University of Chicago Press.

Lakshminarayanan, S. (2010) 'Interoperable Security Standards for Web Services', *IT Professional*, 12 (5), pp. 42-47.

Lee, J.W. & Kim, S.H. (2000) 'Using analytic network process and goal programming for interdependent information system project selection', *Computers and Operations Research*, 27, pp. 367-382.

Lerner, R.M. (2001) 'At the Forge: Introducing SOAP', *Linux Journal*, 2001 (83es), Article 11-March 2001.

Li, P., Hastie, T.J. & Church, K.W. (2007) 'Nonlinear Estimators and Tail Bounds for Dimension Reduction in l1 Using Cauchy Random Projections', *The Journal of Machine Learning Research*, 8, pp. 2497-2532.

Liu, H., Pallickara, S. & Fox, G. (2005) 'Performance of Web Services Security', *Proceedings of the 13th Annual Mardi Gras Conference, Baton Rouge, Louisiana, February 3rd to 5th 2005*.

Liu, M., Guo, Y. & Zhou, L. (2009) 'Text Steganography Based on Online Chat', *Proceedings of the Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2009), Kyoto, Japan, 12-14 September 2009*, pp. 807-810.

Lu, M.H., Madu, C.N., Kuei, C. & Winokur, D. (1994) 'Integrating QFD, AHP and Benchmarking in Strategic Marketing', *Journal of Business & Industrial Marketing*, 82 (2), pp. 250-259.

Machado, A.C.C. & Ferraz, C.A.G. (2005) 'Guidelines for performance evaluation of web services', *Proceedings of the 11th Brazilian Symposium on Multimedia and the Web, WebMedia '05, Pocos de Caldas - Minas Gerais, Brazil*, New York, NY, USA: ACM, pp. 1-10.

Mahmoud, Q.H. (2005) 'Securing Web Services and the Java ESDP 1.5 XWS-Security Framework', 4 January 2006. Available at: http://java.sun.com/developer/technicalArticles/WebServices/security/.

March, S.T. & Smith, G.F. (1995) 'Design and Natural Science Research on Information Technology', *Decision Support Systems*, 15 (4), pp. 251-266.

March, S.T. & Storey, V.C. (2008) 'Design Science in the Information Systems Discipline: An Introduction to the Special Issue on Design Science Research', *Management Information Systems Quarterly*, 32 (4), pp. 725-730.

Markus, M.L., Majchrzak, A. & Gasser, L. (2002) 'A Design Theory for Systems that Support Emergent Knowledge Processes', *Management Information Systems Quarterly*, 26 (3), pp. 179-212.

Memon, A.G., Khawaja, S. & Shah, A. (2008) 'Steganography: A New Horizon for Safe Communication Through XML', *Journal of Theoretical Information Technology*, 4 (3), pp. 187-202.

Menasce, D.A. (2002) 'QoS issues in Web services', *IEEE Internet Computing*, 6 (6), pp. 72-75.

Mi, Z., Zunping, C., Ziji, M. & Binyu, Z. (2005) 'A Security Model Design in Web Service Environment', *Proceedings of the The Fifth International Conference on Computer and Information Technology, CIT '05*, Washington, DC, USA: IEEE Computer Society, pp. 736-740.

Microsoft Corporation (2008) *Comparing Web Service Performance*, Microsoft Corporation. Available at: http://msdn.microsoft.com/en-us/netframework/cc302396.aspx.

Microsoft Corporation (2005) *Web Service Security: Scenarios, Patterns, and Implementation Guidance for Web Services Enhancements*, http://msdn.microsoft.com/en-us/library/aa480613.aspx.

Microsoft Corporation (2004) *Web Services Performance: Comparing Java 2TM Enterprise Edition (J2EETM platform) and the Microsoft® .NET Framework - A Response to Sun Microsystem's Benchmark*, Microsoft Corporation. Available at: http://download.microsoft.com/download/1/9/b/19bc8aa7-05fa-4e86-a612-c2cc181e4ee6/sun_ws_benchmark_response.pdf.

Mingers, J. (2001) 'Combining IS Research Methods: Towards a Pluralist Methodology', *Information Systems Research*, 12 (3), pp. 240-259.

Mitra, N. & Lafon, Y. (2007) *SOAP Version 1.2 Part 0: Primer (Second Edition)*, W3C. Available at: http://www.w3.org/TR/soap12-part0/.

Moralis, A., Pouli, V., Grammatikou, M., Papavassiliou, S. & Maglaris, V. (2007) 'Performance Comparison of Web Services Security: Kerberos Token Profile Against X.509 Token Profile', *Proceedings of the Third International Conference on Networking and Services, ICNS'07*, pp. 28-34.

Mueller, J.P. (2001) *Special Edition Using SOAP*, 1st edn, USA: QUE.

Muller, B. (2010) 15 June 2010-last update, *SOA: from XMLRPC via SOAP to REST*. Available at: http://bobmuller.nl/technologies/soa-esb-and-eai/soa-from-xmlrpc-via-soap-to-rest.

Nadalin, A., Goodner, M., Gudgin, M., Barbir, A. & Granqvist, H. (2007a) *WS-SecureConversation 1.3*, OASIS. Available at: http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.3/ws-secureconversation.html.

Nadalin, A., Goodner, M., Gudgin, M., Barbir, A. & Granqvist, H. (2007b) *WSTrust 1.3*, OASIS. Available at: http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.html.

Nadalin, A., Goodner, M., Gudgin, M., Barbir, A. & Granqvist, H. (2007c) *WS-SecurityPolicy 1.2*, OASIS. Available at: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.html.

Nadalin, A., Kaler, C., Monzillo, R. & Hallam-Baker, P. (2006) *Web Services Security: SOAP Message Security 1.1*, OASIS. Available at: http://docs.oasis-open.org/wss/v1.1/.

Naedele, M. (2003) 'Standards for XML and Web Services Security', *Computer*, 36 (4), pp. 96-98.

Nagappan, R., Skoczylas, R. & Sriganesh, R.P. (2003) *Developing Java Web Services*, Indianapolis, Indiana: Wiley Publishing, Inc.

Nakamur, Y., Hada, S. & Neyama, R. (2002) 'Towards the Integration of Web Services Security on Enterprise Environments', *Proceedings of the 2002 Symposium on Applications and the Internet (SAINT'02w), Nara, Japan, 28 January-1 February 2002*, pp. 166-175.

Nandigam, J., Gudivada, V.N. & Kalavala, M. (2005) 'Semantic Web Services', *J. Comput. Small Coll.*, 21 (1), pp. 50-63.

Netbeans (2011) *Introduction to Web Services.* Oracle Corporation. Available at: http://www.netbeans.com/kb/docs/websvc/intro-ws.html.

Newcomer, E. (2002) *Understanding Web Services: XML, WSDL, SOAP and UDDI*, 1st edn, USA: Addison-Wesley Professional.

Nordbotten, N.A. (2009) 'XML and Web Services Security Standards', *Communications Surveys & Tutorials, IEEE*, 11 (3), pp. 4-21.

Novakouski, M., Simanta, S., Peterson, G., Morris, E.J. & Lewis, G. (2010) *Performance Analysis of WS-Security Mechanisms in SOAP-Based Web Services*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA.

O'Neill, M., Hallam-Baker, P., Cann, S.M., Shema, M., Simon, E., Watters, P.A. & White, A. (2003) *Web Services Security*, 1st edn, New York. USA: McGraw-Hill.

Orlikowski, W.J. & Baroudi, J. (1991) 'Studying information technology in organizations: Research approaches and assumptions', *Information Systems Research*, 2 (1), pp. 1-28.

Owen, C. (1997) 'Design Research: Building the Knowledge Base', *Journal of the Japanese Society for the Science of Design*, 5 (2), pp.36-45.

Por, L.Y. & Delina, B. (2008) 'Information Hiding: A New Approach in Text Steganography', *Proceedings of the 7th International Conference on Applied Computer & Applied Computational Science (ACACOS'08), Hangzhou, China, 6-8 April 2008*, pp. 689-695.

Pries-Heje, J. & Baskerville, R. (2008) 'The Design Theory Nexus', *Management Information Systems Quarterly,* 32 (4), pp. 731-755.

Purao, S. (2002) *Design Research in the Technology of Information Systems: Truth or Dare*, GSU Department of CIS Working Paper, Atlanta.

Ramanathan, R. (1995) 'Using AHP for resource allocation problems', *European Journal of Operational Research,* 80 (2), pp. 410-417.

Rao, Y., Feng, B.Q., Han, J.C. & Li, Z.C. (2004) 'SX-RSRPM: a Security Integrated Model for Web Services', *Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, China, 26-29 August 2004*, pp. 2953-2958.

Ravi, V., Shankar, R. & Tiwari, M.K. (2005) 'Analyzing alternatives in reverse logistics for end-of-life computers: ANP and balanced scorecard approach', *Computers & Industrial Engineering,* 48 (2), pp. 327-356.

Roper-Lowe, G.C. & Sharp, J.A. (1990) 'The Analytic Hierarchy Process and Its Application to an Information Technology Decision', *The Journal of the Operational Research Society,* 41 (1), pp. 49-59.

Saaty, T.L. (2008) 'Decision making with the analytic hierarchy process', *International Journal of Services Sciences,* 1 (1), pp. 83-98.

Saaty, T.L. (2001) *Decision making for leaders : the analytic hierarchy process for decisions in a complex world,* 3rd edn, University of Pittsburgh.

Saaty, T.L. (1990a) 'How to make a decision: The analytic hierarchy process', *European Journal of Operational Research,* 48 (1), pp. 9-26.

Saaty, T.L. (1990b) *Multicriteria decision making: the analytic hierarchy process: planning, priority setting, resource allocation,* Pittsburgh: RWS Publications.

Saaty, T.L. (1987) 'A new macroeconomic forecasting and policy evaluation method using the analytic hierarchy process', *Mathematical Modelling,* 9 (3-5), pp. 219-231.

Saaty, T.L. (1982) *Decision making for leaders : the analytical hierarchy process for decisions in a complex world,* Belmont, Calif: Lifetime Learning Pubns.

Saaty, T.L. & Kearns, K.P. (1985) *Analytical planning,* Pergamon Pres.

Saaty, T.L. & Vargas, L.G. (2000) *Models, methods, concepts and applications of the analytic hierarchy process,* Springer.

Saaty, T.L. & Vargas, L.G. (1991) *Prediction, projection, and forecasting: applications of the analytic hierarchy process in economics, finance, politics, games, and sports,* Kluwer Academic Pub.

Saaty, T.L. & Vargas, L.G. (1984) 'The legitimacy of rank reversal', *Omega*, 12 (5), pp. 513-516.

Salmeron, J.L. & Herrero, I. (2005) 'An AHP-based methodology to rank critical success factors of executive information systems', *Computer Standards and Interfaces*, 28, pp. 1-12.

Santhanam, R. & Kyparisis, G.J. (1996) 'A decision model for interdependent information system project selection', *European Journal of Operational Research*, 89, pp. 380-399.

Shirali-Shahrez, M. (2008) 'Text Steganography by Changing Words Spelling', *Proceedings of the 10th International Conference on Advanced Communication Technology (ICACT 2008), Phoenix Park, Korea, 17-20 February 2008*, pp. 1912-1913.

Shirali-Shahreza, M. (2008) 'A New Synonym Text Steganography', *Proceedings of the 4th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2008), Harbin, China, 15-17 August 2008*, pp. 1524-1526.

Shirasuna, S., Slominski, A., Fang, L. & Gannon, D. (2004) 'Performance comparison of security mechanisms for grid services', *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*, pp. 360-364.

Siddiqui, B. (2003a) 'Web Services Security, Part 1', *Security Articles on webservices.xml.com*, pp. 11th November 2005. Available at: http://webservices.xml.com/pub/a/ws/2003/03/04/security.html.

Siddiqui, B. (2003b) 'Web Services Security, Part 2', *Security Articles on webservices.xml.com*, pp. 11th November 2005. Available at: http://webservices.xml.com/pub/a/ws/2003/04/01/security.html.

Singh, I., Brydon, S., Murray, G., Ramachandran, V., Violleau, T. & Stearns, B. (2004) *Designing Web Services with the J2EETM 1.4 Platform JAX-RPC, SOAP, and XML Technologies*, Santa Clara, California, USA.: Sun Microsystems Inc.

Singhal, A., Winograd, T. & Scarfone, K. (2007) *Guide to Secure Web Services: Recommendations of the National Institute of Standards and Technology*, National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg.

Sosnoski, D.M. (2010) *Java Web services: Metro vs. Axis2 performance*, IBM Corporation. Available at: http://www.ibm.com/developerworks/java/library/j-jws11/.

Sosnoski, D.M. (2009) *Java Web Services: The High Cost of (WS-)Security*, IBM Corporation. Available at: http://www.ibm.com/developerworks/java/library/j-jws6.

Sun Microsystems Inc. (2010) *Metro Users Guide*, Santa Clara, USA: Sun Microsystems Inc. Available at: https://metro.dev.java.net/guide/.

Sun Microsystems Inc. (2007) *The WSIT Tutorial*, Santa Clara, USA: Sun Microsystems Inc. Available at: http://download.oracle.com/docs/cd/E17802_01/webservices/webservices/reference/tutorials/wsit/doc/index.html.

Sun Microsystems Inc. (2004) *Web Services Performance: Comparing JavaTM 2 Enterprise Edition (J2EETM platform) and .NET Framework*, Sun Microsystems Inc. Available at: http://download.microsoft.com/download/1/9/b/19bc8aa7-05fa-4e86-a612-c2cc181e4ee6/sun_ws_benchmark_response.pdf.

Tang, K., Chen, S., Levy, D., Zic, J. & Yan, B. (2006) 'A Performance Evaluation of Web Services Security', *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference, EDOC '06*, pp. 67-74.

Tatsubori, M., Imamura, T. & Nakamura, Y. (2004) 'Best-Practice Patterns and Tool Support for Configuring Secure Web Services Messaging', *Proceeding of the IEEE International Conference on Web Services (ICWS'04), San Diego, California, USA, 6-9 July 2004*, pp. 244-251.

Thirumaran, M., Dhavachelvan, P., Lakshmi, P. & Sheela, S. (2011) 'Parallel analytic hierarchy process for web service discovery and composition', *Proceedings of the 8th International Workshop on Information Integration on the Web: in conjunction with WWW 2011, Hyderabad, India*, New York, NY, USA: ACM, pp. 7:1-7:6.

Ülengin, F. (1994) 'Forecasting foreign exchange rates: A comparative evaluation of AHP', *Omega*, 22 (5), pp. 505-519.

Vaishnavi, V. & Kuechler, W. (2009) August 16th-last update, *Design Science Research in Information Systems.* DESRIST.org. Available at: http://desrist.org/desrist.

Vaughan-Nichols, S. (2002) 7 Jan. 2002-last update, *Fat protocols slow Web services.* ZDNet. Available at: http://www.zdnet.com/news/fat-protocols-slow-web-services/298876.

Vedamuthu, A.S., Orchard, D., Hirsch, F., Hondo, M., Yendluri, P., Boubez, T. & Yalcinalp, U. (2007) *Web Services Policy 1.5 - Framework, W3C Recommendation*, Sun Microsystems. Available at: http://www.w3.org/TR/ws-policy/.

Venkatraman, S., Abraham, A. & Paprzycki, M. (2004) 'Significance of steganography on data security', *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2004), 5-7 April 2004*, pp. 347-351 Vol.2.

W3C (2002a) *XML Encryption Syntax and Processing*. Available at: http://www.w3.org/TR/xmlenc-core/.

W3C (2002b) *XML-Signature Syntax and Processing.* Available at: http://www.w3.org/TR/xmldsig-core/.

W3C (2001) *XML Key Management Specification (XKMS).* Available at: http://www.w3.org/TR/xkms/.

Wang, H., Huang, J.Z., Qu, Y. & Xie, J. (2004) 'Web Services: Problems and Future Directions', *Journal of Web Semantics,* 1, pp. 309-320.

Wei, C.C., Chien, C.F. & Wang, M.J.J. (2005) 'An AHP-based approach to ERP system selection', *International Journal of Production Economics,* 96 (1), pp. 47-62.

Wu, C. & Chang, E. (2007) 'Intelligent Web Services Selection based on AHP and Wiki', *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence,* pp. 767-770.

Yang, A. (2002) 'Web Services Security', *EAI Journal*, September (1), pp. 19-23.

Zeng, G., Jiang, R., Huang, G., Xu, M. & Li, J. (2007) 'Optimization of wastewater treatment alternative selection by hierarchy grey relational analysis', *Journal of environmental management,* 82 (2), pp. 250-259.

Zhang, X., Wang, H. & Sun, J. (2007) 'An Information Hiding Method based on SOAP', *Proceedings of the Third International Conference on International Information Hiding and Multimedia Signal Processing (IIH-MSP 2007),* Washington, DC, USA: IEEE Computer Society, pp. 453-456.

Zimmerman, J. (2007) 31 January 2007-last update, *SOAP, XML-RPC and REST.* Wordpress. Available at: http://jimmyzimmerman.com/blog/2007/01/soap-xml-rpc-and-rest.html.

Zuo, M., Wang, S. & Wu, B. (2008) 'Research on Web services selection model based on AHP', *Proceedings of IEEE International Conference on Service Operations and Logistics, and Informatics, IEEE/SOLI 2008*, pp. 2763-2768.

Appendix A

Examples of SOAP Messages Based on a One Character Initial Message

Profile	**Number of SOAP requests**	**Request length (Bytes)**	**Number of SOAP responses**	**Response length (Bytes)**
Simple (No Security)	1	205	1	217
UA	1	7693	1	5345
MCS	1	7285	1	5877
Reliable Messaging	4	5674 (1222+1672+1144+1636)	4	5710 (1222+2143+777+1568)

Example 1: Simple WS (No Security)

	Request	Response
1	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Body> <ns2:sendEcho xmlns:ns2="http://s00r0.server.bash/"> <echoText>**A**</echoText> </ns2:sendEcho> </S:Body> </S:Envelope>	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Body> <ns2:sendEchoResponse xmlns:ns2="http://s00r0.server.bash/"> <return>**A**</return> </ns2:sendEchoResponse> </S:Body> </S:Envelope>

Example 2: Username Authentication with Symmetric Key (UA)

	Request	Response
1	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:exc14n="http://www.w3.org/2001/10/xml-exc-c14n#" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"> <S:Header> <To wsu:Id="_5006" xmlns="http://www.w3.org/2005/08/addressing">http://cspgbba_desk.disc.stjohns.brunel.ac.uk:4040/EchoWebApplication/EchoS01R0WSService</To> <Action wsu:Id="_5005" xmlns="http://www.w3.org/2005/08/addressing">http://s01r0.server.bash/EchoS01R0WS/sendEchoRequest</Action> <ReplyTo wsu:Id="_5004" xmlns="http://www.w3.org/2005/08/addressing"> <Address>http://www.w3.org/2005/08/addressing/anonymous</Address> </ReplyTo> <MessageID wsu:Id="_5003" xmlns="http://www.w3.org/2005/08/addressing">uuid:f780f110-9efd-47ed-aa59-e5acc61368b5</MessageID> <wsse:Security S:mustUnderstand="1"> <wsu:Timestamp wsu:Id="_3" xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"> <wsu:Created>2008-11-02T23:50:35Z</wsu:Created> <wsu:Expires>2008-11-02T23:55:35Z</wsu:Expires> </wsu:Timestamp> <xenc:EncryptedKey Id="_5002" xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"> <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/> <ds:KeyInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="keyInfo"> <wsse:SecurityTokenReference> <wsse:KeyIdentifier EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509SubjectKeyIdentifier">dVE29ysyFW/iD1la3ddePzM6IWo=</wsse:KeyIdentifier> </wsse:SecurityTokenReference> </ds:KeyInfo> <xenc:CipherData> <xenc:CipherValue>E9460LKVK7uQog7QdWhlOp2Bb/ffPoCIez3J4UhSbriSZ3Ds4xB9/MvJj+sJSIeai7ZztzrML9dEQfU8Ov43bTNX+Jj5vvsJNN94TdSD9Oa0wGsbqLvnZIDQgxWNo7XFlrq8r5zzp2aU141pcPEpRbQuYdpTcu0CFDlMtu5B1g0=</xenc:CipherValue> </xenc:CipherData> </xenc:EncryptedKey> <xenc:ReferenceList xmlns="" xmlns:ns16="http://www.w3.org/2003/05/soap-envelope" xmlns:ns17="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"> <xenc:DataReference URI="#_5008"/> <xenc:DataReference URI="#_5009"/> </xenc:ReferenceList> <xenc:EncryptedData Id="_5009" Type="http://www.w3.org/2001/04/xmlenc#Element"	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:exc14n="http://www.w3.org/2001/10/xml-exc-c14n#" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"> <S:Header> <To wsu:Id="_5005" xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/anonymous</To> <Action wsu:Id="_5003" xmlns="http://www.w3.org/2005/08/addressing">http://s01r0.server.bash/EchoS01R0WS/sendEchoResponse</Action> <MessageID wsu:Id="_5002" xmlns="http://www.w3.org/2005/08/addressing">uuid:09bc6aba-46eb-422d-871c-f5a97d63a528</MessageID> <RelatesTo wsu:Id="_5004" xmlns="http://www.w3.org/2005/08/addressing">uuid:f780f110-9efd-47ed-aa59-e5acc61368b5</RelatesTo> <wsse:Security S:mustUnderstand="1"> <wsu:Timestamp wsu:Id="_3" xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"> <wsu:Created>2008-11-02T23:50:37Z</wsu:Created> <wsu:Expires>2008-11-02T23:55:37Z</wsu:Expires> </wsu:Timestamp> <xenc:ReferenceList xmlns="" xmlns:ns16="http://www.w3.org/2003/05/soap-envelope" xmlns:ns17="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"> <xenc:DataReference URI="#_5007"/> </xenc:ReferenceList> <ds:Signature Id="_1" xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"> <ds:SignedInfo> <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"> <exc14n:InclusiveNamespaces PrefixList="wsse S"/> </ds:CanonicalizationMethod> <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/> <ds:Reference URI="#_5002"> <ds:Transforms> <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"> <exc14n:InclusiveNamespaces PrefixList="S"/> </ds:Transform> </ds:Transforms> <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/> <ds:DigestValue>leCh8+B0DZ9FV73NIZBEGKZmeGw=</ds:DigestValue> </ds:Reference> <ds:Reference URI="#_5003"> <ds:Transforms> <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"> <exc14n:InclusiveNamespaces PrefixList="S"/> </ds:Transform> </ds:Transforms> <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/> <ds:DigestValue>JyDAngE0Rm2fAfB7WIYDJ5eG8kI=</ds:DigestValue> </ds:Reference>

```
        xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
        <ds:KeyInfo
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="keyInfo">
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#_5002" ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-
1.1#EncryptedKey"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>

<xenc:CipherValue>3di4cQ/jAcVfnkgw2vljdWDwns7AxpVRJq+D/DXeouWcuxLNSDxKsAmQYfE+pXjzAUZRyaq9V0rsKbHrvQRwM1S6b
9dUf2WLnsgZOJ7CboMkg3dtWgqneYS0luOikzgf3BO2b/gZMatnS6/FJpCosHQeB1D046kF1ru2fN2GSejNa1fk26/Qu8LWjnQFk9Vj10qlHuY
gb2/SprJulH5mgEhOx4PLv2+GS6L9xdTIslpfV5RTtjoIrp8/BHb9mhfEKRfMqCppfFCOyDga3GX9fIY7GkeYuDPdOyTRJ0Es201Aw0byUBlz
SQWtdqkkL7q8vRnUc8uhDDB/A7CrU0lNxGwrTyh9YUwNjn1mFHSGohBy0VHRiFAF0y0+uj1aadCWvCgEsvsdseYlqBr+OtTD2sip/y1mYP
Vsre2xe5obXmJzFXzJriQrcZz208MjMOE5Vvz+U0PY/RRUqfXQdtkUxQnGs/T01yh7+E3R3LMCrhk3w+PISuNLCSPG0Sf6lrUA</xenc:Ciph
erValue>
        </xenc:CipherData>
      </xenc:EncryptedData>
      <ds:Signature Id="_1"
        xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <exc14n:InclusiveNamespaces PrefixList="wsse S"/>
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
          <ds:Reference URI="#_5003">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <exc14n:InclusiveNamespaces PrefixList="S"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>AiAgXvnVHF6eatthuPwzG4A/hw8=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#_5004">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <exc14n:InclusiveNamespaces PrefixList="S"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>5Ab1ebo4/FraGgck/A8iDx1J9+I=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#_5005">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <exc14n:InclusiveNamespaces PrefixList="S"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>cpHaSaANqa1Ctogh/9Gv6rbbivs=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#_5006">
            <ds:Transforms>
          <ds:Reference URI="#_5004">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <exc14n:InclusiveNamespaces PrefixList="S"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>pOE9k50Odm1bpagrnYxikFnj8IQ=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#_5005">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <exc14n:InclusiveNamespaces PrefixList="S"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>Nd/8wVmBdLowQKMblBRYK+6xcjA=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#_5006">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <exc14n:InclusiveNamespaces PrefixList="S"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>6QWqKtiSnibxN8alXBJ+aUy1yVA=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#_3">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <exc14n:InclusiveNamespaces PrefixList="wsu wsse S"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>Dji/v8LpUM1zccDNhJsHd6fVaH8=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>nUNYjCespsZKXbVKWnE0TdkFVWA=</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier
              EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-
1.1#EncryptedKeySHA1">uVSGWduakDJe2qHFnEZ4dUt/ljU=</wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id="_5006">
    <xenc:EncryptedData Id="_5007"
      Type="http://www.w3.org/2001/04/xmlenc#Content"
      xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
      <ds:KeyInfo
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="keyInfo">
```

```
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <exc14n:InclusiveNamespaces PrefixList="S"/>
                    </ds:Transform>
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>BLoJ2F/86smJfhXz7WZzid7Dx4Y=</ds:DigestValue>
            </ds:Reference>
            <ds:Reference URI="#_5007">
                <ds:Transforms>
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <exc14n:InclusiveNamespaces PrefixList="S"/>
                    </ds:Transform>
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>LX/ljKbElCK0/l0OY6BTLEW1WVk=</ds:DigestValue>
            </ds:Reference>
            <ds:Reference URI="#_3">
                <ds:Transforms>
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <exc14n:InclusiveNamespaces PrefixList="wsu wsse S"/>
                    </ds:Transform>
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>Kzo9PEsY7D3nnraB42PZO+XbG+U=</ds:DigestValue>
            </ds:Reference>
            <ds:Reference URI="#uuid_3aed497d-ff1c-4a37-964e-56f9189d080c">
                <ds:Transforms>
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <exc14n:InclusiveNamespaces PrefixList="wsu wsse S"/>
                    </ds:Transform>
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>XbDgClGFz2z0OMkdeDN4xlMrMEo=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>4RPUOnA4IDzOyMG/ZwGJ9TAhC+o=</ds:SignatureValue>
        <ds:KeyInfo>
            <wsse:SecurityTokenReference wsu:Id="uuid_c1149094-08fa-49ea-996f-a8ea9ce37b56">
                <wsse:Reference URI="#_5002" ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-
1.1#EncryptedKey"/>
            </wsse:SecurityTokenReference>
        </ds:KeyInfo>
    </ds:Signature>
</wsse:Security>
</S:Header>
<S:Body wsu:Id="_5007">
    <xenc:EncryptedData Id="_5008"
        Type="http://www.w3.org/2001/04/xmlenc#Content"
        xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
        <ds:KeyInfo
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="keyInfo">
            <wsse:SecurityTokenReference>
                <wsse:Reference URI="#_5002" ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-
1.1#EncryptedKey"/>
            </wsse:SecurityTokenReference>
                <wsse:SecurityTokenReference>
                    <wsse:KeyIdentifier
                        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/oasis-wss-soap-message-security-
1.1#EncryptedKeySHA1">uVSGWduakDJe2qHFnEZ4dUt/ljU=</wsse:KeyIdentifier>
                </wsse:SecurityTokenReference>
            </ds:KeyInfo>
            <xenc:CipherData>

<xenc:CipherValue>rnfTD8vqOU7YuMiZKbuvxath4/uStxTsxh/yuGVmvvC+OKzNAKOzHFxBTp/hIgHNl0eAGrJHJ7bD2/xuM6wnLMidQO
UyZ1nEOtATqOWPAl7VVauurHg48JxE2WzEcbVKpLK/XaeyOyUOQFjXvljqHuc8dPfGlZ1GM+wQ0Ggwvpg=</xenc:CipherValue>
            </xenc:CipherData>
        </xenc:EncryptedData>
    </S:Body>
</S:Envelope>
```

	</ds:KeyInfo> <xenc:CipherData> <xenc:CipherValue>KVFK0+8PB54BvOq1oT2AoC0JV6odWJX+CH0quyzD2lRd08Gf4Qd+0r2f9xdMOjVeWGqPmCP618PwK4X5RNjW68IN4FOI4S0Rv4/qpc65ygF49fgDRy4hwsYay97QQ6/4n/22a5rFjf9sQW93Tb99dQ==</xenc:CipherValue> </xenc:CipherData> </xenc:EncryptedData> </S:Body> </S:Envelope>	

Example 3: Mutual Certificates (MCS)

	Request	Response
1	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:exc14n="http://www.w3.org/2001/10/xml-exc-c14n#" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"> <S:Header> <To wsu:Id="_5005" xmlns="http://www.w3.org/2005/08/addressing">http://cspgbba_desk.disc.stjohns.brunel.ac.uk:4040/EchoWebApplication/EchoS02R0WSService</To> <Action wsu:Id="_5004" xmlns="http://www.w3.org/2005/08/addressing">http://s02r0.server.bash/EchoS02R0WS/sendEchoRequest</Action> <ReplyTo wsu:Id="_5003" xmlns="http://www.w3.org/2005/08/addressing"> <Address>http://www.w3.org/2005/08/addressing/anonymous</Address> </ReplyTo> <MessageID wsu:Id="_5002" xmlns="http://www.w3.org/2005/08/addressing">uuid:28076cd2-4ceb-40d6-ae9c-6de83f5b03c6</MessageID> <wsse:Security S:mustUnderstand="1"> <wsu:Timestamp wsu:Id="_3" xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"> <wsu:Created>2008-11-02T23:53:05Z</wsu:Created> <wsu:Expires>2008-11-02T23:58:05Z</wsu:Expires> </wsu:Timestamp> <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="uuid_a18fed82-16fb-4ba7-addb-bb886f30134f" xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512">MIIDDzCCAnigAwIBAgIBAzANBgkqhkiG9w0BAQQFADBOMQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEMMAoGA1UEChMDU1VOMQwwCgYDVQQLEwNKV1MxDjAMBgNVBAMTBVNVTkNBMB4XDTA3MDMxMjEwMjQ0MFoXDTE3MDMwOTEwMjQ0MFowbzELMAkGA1UEBhMCQVUxEzARBgNVBAgTClNvbWUtU3RhdGUxITAfBgNVBAoTGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZDEMMAoGA1UECxMDU1VOMRowGAYDVQQDExF4d3NzZWN1cml0eWNsaWVudDCBnz	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:exc14n="http://www.w3.org/2001/10/xml-exc-c14n#" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"> <S:Header> <To wsu:Id="_5005" xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/anonymous</To> <Action wsu:Id="_5003" xmlns="http://www.w3.org/2005/08/addressing">http://s02r0.server.bash/EchoS02R0WS/sendEchoResponse</Action> <MessageID wsu:Id="_5002" xmlns="http://www.w3.org/2005/08/addressing">uuid:b922c13e-e733-4f08-bab2-a92c34e88b8c</MessageID> <RelatesTo wsu:Id="_5004" xmlns="http://www.w3.org/2005/08/addressing">uuid:28076cd2-4ceb-40d6-ae9c-6de83f5b03c6</RelatesTo> <wsse:Security S:mustUnderstand="1"> <wsu:Timestamp wsu:Id="_3" xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"> <wsu:Created>2008-11-02T23:53:05Z</wsu:Created> <wsu:Expires>2008-11-02T23:58:05Z</wsu:Expires> </wsu:Timestamp> <xenc:EncryptedKey Id="_5007" xmlns:ns10="http://www.w3.org/2003/05/soap-envelope" xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"> <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/> <ds:KeyInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="keyInfo"> <wsse:SecurityTokenReference> <wsse:KeyIdentifier EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509SubjectKeyIdentifier">/mItfvuFdS7A0GCysE71TFRxP2c=</wsse:KeyIdentifier> </wsse:SecurityTokenReference> </ds:KeyInfo>

```
ANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAvYxVZKIzVdGMSBkW4bYnV80MV/RgQKV1bf/DoMTX8laMO45P6rlEarxQiOYrgzuYp
+snzz2XM0S6o3JGQtXQuzDwcwPkH55bHFwHgtOMzxG4SQ653a5Dzh04nsmJvxvbncNH/XNaWfHaC0JHBEfNCMwRebYocxYM92pq/G5
OGyECAwEAAaOB2zCB2DAJBgNVHRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZpY2F0
ZTAdBgNVHQ4EFgQU/mItfvuFdS7A0GCysE71TFRxP2cwfgYDVR0jBHcwdYAUZ7plxs6VyOOOTSFyojDV0/YYjJWhUqRQME4xCzAJB
gNVBAYTAkFVMRMwEQYDVQQIEwpTb21lLVN0YXRlMQwwCgYDVQQKEwNTVU4xDDAKBgNVBAsTA0pXUzEOMAwGA1UEAx
MFU1VOQ0GCCQDbHkJaq6KijjANBgkqhkiG9w0BAQQFAAOBgQBEnRdcQeMyCYqOHw2jbPOPUlvu07bZe7sI3ly/Qz+4mkrFctqMSupgh
QtLv9dZcqDOUFLCGMse7+l5MG00VawzsoVe242iXzJB111ePzhhppIPOHXXtflj/JD2U4Qz75C/dfdd5AAZbqGSFtZh7pyE8Ot1vOq7R48/bH
uvTsEVUQ==</wsse:BinarySecurityToken>
            <xenc:EncryptedKey Id="_5007"
                xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"                                        xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512">
                <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
                <ds:KeyInfo
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="keyInfo">
                    <wsse:SecurityTokenReference>
                        <wsse:KeyIdentifier
                            EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509SubjectKeyIdentifier">dVE29ysyFW/iD1la3ddePzM6IWo=</wsse:KeyIdentifier>
                    </wsse:SecurityTokenReference>
                </ds:KeyInfo>
                <xenc:CipherData>

<xenc:CipherValue>mYictXCXb4po/chiQP+x6+UTZv3ZFuIccHYzOx9J6M1HPSmWYLDEUE8EkKU4W6YTU8Y0uopjlH6Dgsun+wqItDU
mKj0Yg3whFCNWlXvvltS9conN6J3KSZTz85zK1LY8mAmcSTTegplRlsqCtuPUYNWV7/+1br4+JBUSgOfPGy0=</xenc:CipherValue>
                </xenc:CipherData>
                <xenc:ReferenceList>
                    <xenc:DataReference URI="#_5008"/>
                </xenc:ReferenceList>
            </xenc:EncryptedKey>
            <ds:Signature Id="_1"
                xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"                                        xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512">
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <exc14n:InclusiveNamespaces PrefixList="wsse S"/>
                    </ds:CanonicalizationMethod>
                    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
```

```
                <xenc:CipherData>

<xenc:CipherValue>l8HHf0gE3XIqazuL/U2x/bZOC8t0NC5fs04Q/uq5RU9R6AT9iPTEb9i/ab/3LNFwnUFy67m3zfQmbMuKeaE+otneBEEHX
8K6m0vjNfG7Hc2wIRgQO2Ef8MDQOaaMFP1ZKSoK9bLXo9xD+hQ1jXCtCUNL1VaiEJgcxAOtDl/uOfI=</xenc:CipherValue>
                </xenc:CipherData>
                <xenc:ReferenceList>
                    <xenc:DataReference URI="#_5008"/>
                </xenc:ReferenceList>
            </xenc:EncryptedKey>
            <ds:Signature Id="_1"
                xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"                                        xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512">
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <exc14n:InclusiveNamespaces PrefixList="wsse S"/>
                    </ds:CanonicalizationMethod>
                    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                    <ds:Reference URI="#_5002">
                        <ds:Transforms>
                            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                                <exc14n:InclusiveNamespaces PrefixList="S"/>
                            </ds:Transform>
                        </ds:Transforms>
                        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>fZevxFpoUe0O+ib+4cJ5c/5QtAI=</ds:DigestValue>
                    </ds:Reference>
                    <ds:Reference URI="#_5003">
                        <ds:Transforms>
                            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                                <exc14n:InclusiveNamespaces PrefixList="S"/>
                            </ds:Transform>
                        </ds:Transforms>
                        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>5tMtOXddKS5hpu/EssMo5gLSMGY=</ds:DigestValue>
                    </ds:Reference>
                    <ds:Reference URI="#_5004">
                        <ds:Transforms>
                            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
```

```
        <ds:Reference URI="#_5002">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <exc14n:InclusiveNamespaces PrefixList="S"/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>c2r5I0nidByse48vZ1s5K7gVX4E=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#_5003">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <exc14n:InclusiveNamespaces PrefixList="S"/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>bfaj3tu9jIeOXTrb3JWtYD+ZKaI=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#_5004">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <exc14n:InclusiveNamespaces PrefixList="S"/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>BAhm4BLMooiBr1NI9rwLm+1OYY4=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#_5005">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <exc14n:InclusiveNamespaces PrefixList="S"/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>X2+HYlWW9NOqjhw54V+HbcSKkDM=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#_5006">
          <ds:Transforms>
              <exc14n:InclusiveNamespaces PrefixList="S"/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>+YMQSKO6KUAZZiZRNK2Aw/X5IyU=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#_5005">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <exc14n:InclusiveNamespaces PrefixList="S"/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>Nd/8wVmBdLowQKMblBRYK+6xcjA=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#_5006">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <exc14n:InclusiveNamespaces PrefixList="S"/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>K56DINu9+dyLCrPwOAmcTiV1fZU=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#_3">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
              <exc14n:InclusiveNamespaces PrefixList="wsu wsse S"/>
            </ds:Transform>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>X9xUde9yklabvg1xhmtQscPwbkw=</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>

<ds:SignatureValue>bpomFxcblaE+EMDNt6RRaXAmYkZTKTK94tmzlvq0WnXni5iBcMXpBP5elLM+b87a8NWQ8IHdlIEaz3/r0wOOmz3fh
wiSiEOIaoN9BR8iGziUeZ3jzjbfIBdeq7pQr9MK8j9MELgto4MSotim/csImXLrE3Y+O1cX7OomIfb3sNU=</ds:SignatureValue>
      <ds:KeyInfo>
```

```
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <exc14n:InclusiveNamespaces PrefixList="S"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>UOc+OFH9SsG+7m2KsX0TO4y4unQ=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#_3">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <exc14n:InclusiveNamespaces PrefixList="wsu wsse S"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>X9xUde9yklabvg1xhmtQscPwbkw=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>

<ds:SignatureValue>YjI9Ee0jJOZkepquOFmUigXb4Ms1VlDEj7bqzvMvIsSk42FAFkHQrteiqkLcZqym/NEsuZGbuneTf8384KEl76Ik67EXT9
R4yLwS6v5CD5TUmYmvGTjrM0sy3+JN8khG3QoXgqST7vKLycfPYRzlA7PkdsNb7MG5QN+KYBdvypY=</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference
              URI="#uuid_a18fed82-16fb-4ba7-addb-bb886f30134f"        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id="_5006">
    <xenc:EncryptedData Id="_5008"
      Type="http://www.w3.org/2001/04/xmlenc#Content"
      xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"                        xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
      <xenc:CipherData>
```

```
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier
              EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509SubjectKeyIdentifier">dVE29ysyFW/iD1la3ddePzM6IWo=</wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id="_5006">
    <xenc:EncryptedData Id="_5008"
      Type="http://www.w3.org/2001/04/xmlenc#Content"
      xmlns:ns10="http://www.w3.org/2003/05/soap-envelope"                        xmlns:ns11="http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
      <xenc:CipherData>

<xenc:CipherValue>Dq6GxW9E98czYA6sGGGnCDSUal5+t5moWseIth61pp1wkKbGNdFOTMvHoLTE82btl/+01KHhkEUV8G8bhvbyklXt6
+8rPQkNChbrCP2cQENN+uEo8TTN2lxcs2fetBU1W8ddR+BZrjhzo1jA1y4h5OGJ25M0592I0tzuF0DLXuA=</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

	<xenc:CipherValue>KX4UCZjxjepkkAHbJDsCt2UbXvLNXODxLUD8AU1axYGV/hoPjo1fFjuDu94m6XvcoyqwKfXVZKMMmiklMddg+hmR+wK1T3tUY+TtuP/tmd/bNFQkSmoN+RfU0LfAx8DaNnqvuz9QJt1Ujm3mEQqbjg==</xenc:CipherValue>
	          </xenc:CipherData>
	      </xenc:EncryptedData>
	   </S:Body>
	</S:Envelope>	

Example 4: Reliable Messaging

	Request	Response

1	<?xml version="1.0" encoding="UTF-8"?>
	<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
	  <S:Header>
	    <To xmlns="http://www.w3.org/2005/08/addressing">http://cspgbba_desk.disc.stjohns.brunel.ac.uk:4040/EchoWebApplication/EchoS00R1WSService</To>
	    <Action xmlns="http://www.w3.org/2005/08/addressing">http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequence</Action>
	    <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">
	      <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
	    </ReplyTo>
	    <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:5e9caaa4-c296-4972-b9c1-8910e573978f</MessageID>
	  </S:Header>
	  <S:Body>
	    <ns2:CreateSequence
	      xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm"
	      xmlns:ns3="http://www.w3.org/2005/08/addressing"
	      xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
	      xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
	xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm">
	      <ns2:AcksTo>
	        <ns3:Address>http://www.w3.org/2005/08/addressing/anonymous</ns3:Address>
	      </ns2:AcksTo>
	      <ns2:Offer>
	        <ns2:Identifier>uuid:559afeed-9662-400c-9bc2-e15c2443c98f</ns2:Identifier>
	      </ns2:Offer>
	    </ns2:CreateSequence>
	  </S:Body>
	</S:Envelope>
		<?xml version="1.0" encoding="UTF-8"?>
		<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
		  <S:Header>
		    <To xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/anonymous</To>
		    <Action xmlns="http://www.w3.org/2005/08/addressing">http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequenceResponse</Action>
		    <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:3416b50a-2bbc-49ed-90e5-f221b11ada10</MessageID>
		    <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:5e9caaa4-c296-4972-b9c1-8910e573978f</RelatesTo>
		  </S:Header>
		  <S:Body>
		    <ns2:CreateSequenceResponse
		      xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm"
		      xmlns:ns3="http://www.w3.org/2005/08/addressing"
		      xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
		      xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
		xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm">
		      <ns2:Identifier>uuid:a0c16e3b-523e-47db-af6e-76233d7bbe0d</ns2:Identifier>
		      <ns2:Accept>
		        <ns2:AcksTo>
		<ns3:Address>http://cspgbba_desk.disc.stjohns.brunel.ac.uk:4040/EchoWebApplication/EchoS00R1WSService</ns3:Address>
		        </ns2:AcksTo>
		      </ns2:Accept>
		    </ns2:CreateSequenceResponse>
		  </S:Body>
		</S:Envelope>

2	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Header> <To xmlns="http://www.w3.org/2005/08/addressing">http://cspgbba_desk.disc.stjohns.brunel.ac.uk:4040/EchoWebApplication/EchoS00R1WSService</To> <Action xmlns="http://www.w3.org/2005/08/addressing">http://s00r1.server.bash/EchoS00R1WS/sendEchoRequest</Action> <ReplyTo xmlns="http://www.w3.org/2005/08/addressing"> <Address>http://www.w3.org/2005/08/addressing/anonymous</Address> </ReplyTo> <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:a4129c51-bfa1-4e96-9933-5b8b39ae4afe</MessageID> <ns2:Sequence xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm" xmlns:ns3="http://www.w3.org/2005/08/addressing" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm"> <ns2:Identifier>uuid:a0c16e3b-523e-47db-af6e-76233d7bbe0d</ns2:Identifier> <ns2:MessageNumber>1</ns2:MessageNumber> </ns2:Sequence> <ns2:AckRequested xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm" xmlns:ns3="http://www.w3.org/2005/08/addressing" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm"> <ns2:Identifier>uuid:a0c16e3b-523e-47db-af6e-76233d7bbe0d</ns2:Identifier> </ns2:AckRequested> </S:Header> <S:Body> <ns2:sendEcho xmlns:ns2="http://s00r1.server.bash/"> <echoText>A</echoText> </ns2:sendEcho> </S:Body> </S:Envelope>	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Header> <To xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/anonymous</To> <Action xmlns="http://www.w3.org/2005/08/addressing">http://s00r1.server.bash/EchoS00R1WS/sendEchoResponse</Action> <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:b973c467-e39f-4061-8d8c-07312d279191</MessageID> <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:a4129c51-bfa1-4e96-9933-5b8b39ae4afe</RelatesTo> <ns2:Sequence xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm" xmlns:ns3="http://www.w3.org/2005/08/addressing" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm"> <ns2:Identifier>uuid:559afeed-9662-400c-9bc2-e15c2443c98f</ns2:Identifier> <ns2:MessageNumber>1</ns2:MessageNumber> </ns2:Sequence> <ns2:AckRequested xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm" xmlns:ns3="http://www.w3.org/2005/08/addressing" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm"> <ns2:Identifier>uuid:559afeed-9662-400c-9bc2-e15c2443c98f</ns2:Identifier> </ns2:AckRequested> <ns2:SequenceAcknowledgement xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm" xmlns:ns3="http://www.w3.org/2005/08/addressing" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm"> <ns2:Identifier>uuid:a0c16e3b-523e-47db-af6e-76233d7bbe0d</ns2:Identifier> <ns2:AcknowledgementRange Lower="1" Upper="1"/> </ns2:SequenceAcknowledgement> </S:Header> <S:Body>

		<ns2:sendEchoResponse xmlns:ns2="http://s00r1.server.bash/"> <return>A</return> </ns2:sendEchoResponse> </S:Body> </S:Envelope>
3	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Header> <ns2:Sequence xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm" xmlns:ns3="http://www.w3.org/2005/08/addressing" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm"> <ns2:Identifier>uuid:a0c16e3b-523e-47db-af6e-76233d7bbe0d</ns2:Identifier> <ns2:MessageNumber>2</ns2:MessageNumber> <ns2:LastMessage/> </ns2:Sequence> <To xmlns="http://www.w3.org/2005/08/addressing">http://cspgbba_desk.disc.stjohns.brunel.ac.uk:4040/EchoWebApplication/EchoS00R1WSService</To> <Action xmlns="http://www.w3.org/2005/08/addressing">http://schemas.xmlsoap.org/ws/2005/02/rm/LastMessage</Action> <ReplyTo xmlns="http://www.w3.org/2005/08/addressing"> <Address>http://www.w3.org/2005/08/addressing/anonymous</Address> </ReplyTo> <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:fa473ac0-a927-4075-96b5-45d84a5aa03e</MessageID> </S:Header> <S:Body/> </S:Envelope>	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Header> <ns2:SequenceAcknowledgement xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm" xmlns:ns3="http://www.w3.org/2005/08/addressing" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm"> <ns2:Identifier>uuid:a0c16e3b-523e-47db-af6e-76233d7bbe0d</ns2:Identifier> <ns2:AcknowledgementRange Lower="1" Upper="2"/> </ns2:SequenceAcknowledgement> <Action xmlns="http://www.w3.org/2005/08/addressing">http://schemas.xmlsoap.org/ws/2005/02/rm/LastMessage</Action> </S:Header> <S:Body/> </S:Envelope>
4	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Header> <ns2:SequenceAcknowledgement xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm" xmlns:ns3="http://www.w3.org/2005/08/addressing" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm">	<?xml version="1.0" encoding="UTF-8"?> <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Header> <To xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/anonymous</To> <Action xmlns="http://www.w3.org/2005/08/addressing">http://schemas.xmlsoap.org/ws/2005/02/rm/TerminateSequence</Action> <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:0d92d0cb-43a7-4fd9-95f8-fdc13ecbd40b</MessageID>

	<ns2:Identifier>uuid:559afeed-9662-400c-9bc2-e15c2443c98f</ns2:Identifier> <ns2:AcknowledgementRange Lower="1" Upper="1"/> </ns2:SequenceAcknowledgement> <To xmlns="http://www.w3.org/2005/08/addressing">http://cspgbba_desk.disc.stjohns.brunel.ac.uk:4040/EchoWebApplication/EchoS00R1WSService</To> <Action xmlns="http://www.w3.org/2005/08/addressing">http://schemas.xmlsoap.org/ws/2005/02/rm/TerminateSequence</Action> <ReplyTo xmlns="http://www.w3.org/2005/08/addressing"> <Address>http://www.w3.org/2005/08/addressing/anonymous</Address> </ReplyTo> <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:eb4cd190-e4cc-46da-a17f-31a72a2131cc</MessageID> </S:Header> <S:Body> <ns2:TerminateSequence xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm" xmlns:ns3="http://www.w3.org/2005/08/addressing" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm"> <ns2:Identifier>uuid:a0c16e3b-523e-47db-af6e-76233d7bbe0d</ns2:Identifier> </ns2:TerminateSequence> </S:Body> </S:Envelope>	<RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:eb4cd190-e4cc-46da-a17f-31a72a2131cc</RelatesTo> <ns2:SequenceAcknowledgement xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm" xmlns:ns3="http://www.w3.org/2005/08/addressing" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm"> <ns2:Identifier>uuid:a0c16e3b-523e-47db-af6e-76233d7bbe0d</ns2:Identifier> <ns2:AcknowledgementRange Lower="1" Upper="2"/> </ns2:SequenceAcknowledgement> </S:Header> <S:Body> <ns2:TerminateSequence xmlns:ns2="http://schemas.xmlsoap.org/ws/2005/02/rm" xmlns:ns3="http://www.w3.org/2005/08/addressing" xmlns:ns4="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ns5="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:ns6="http://schemas.microsoft.com/ws/2006/05/rm"> <ns2:Identifier>uuid:559afeed-9662-400c-9bc2-e15c2443c98f</ns2:Identifier> </ns2:TerminateSequence> </S:Body> </S:Envelope>

Appendix B

AHP Matrices

Pairwise Comparison Matrix for the Alternatives with Respect to Peer Authentication Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	0.33	0.33	0.33	0.33	0.33
UDP	1	1	0.33	0.33	0.33	0.33	0.33
MCS	3	3	1	1	1	1	1
SSL	3	3	1	1	1	1	1
SA	3	3	1	1	1	1	1
SV	3	3	1	1	1	1	1
STS	3	3	1	1	1	1	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.0588	0.0588	0.1765	0.1765	0.1765	0.1765	0.1765

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to Message Origin Authentication Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	1	9	9	9	1
UDP	1	1	1	9	9	9	1
MCS	1	1	1	9	9	9	1
SSL	0.11	0.11	0.11	1	1	1	0.11
SA	0.11	0.11	0.11	1	1	1	0.11
SV	0.11	0.11	0.11	1	1	1	0.11
STS	1	1	1	9	9	9	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.2308	0.2308	0.2308	0.0256	0.0256	0.0256	0.2308

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to Message Integrity Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	1	1	1	1	1
UDP	1	1	1	1	1	1	1
MCS	1	1	1	1	1	1	1
SSL	1	1	1	1	1	1	1
SA	1	1	1	1	1	1	1
SV	1	1	1	1	1	1	1
STS	1	1	1	1	1	1	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.1429	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to Message Confidentiality Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	0.33	0.33	0.33	0.33	0.33
UDP	1	1	0.33	0.33	0.33	0.33	0.33
MCS	3	3	1	1	1	1	1
SSL	3	3	1	1	1	1	1
SA	3	3	1	1	1	1	1
SV	3	3	1	1	1	1	1
STS	3	3	1	1	1	1	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.0588	0.0588	0.1765	0.1765	0.1765	0.1765	0.1765

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to Nonrepudiation Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	1	1	1	1	1
UDP	1	1	1	1	1	1	1
MCS	1	1	1	1	1	1	1
SSL	1	1	1	1	1	1	1
SA	1	1	1	1	1	1	1
SV	1	1	1	1	1	1	1
STS	1	1	1	1	1	1	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.1429	0.1429	0.1429	0.1429	0.1429	0.1429	0.1429

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to Authorization Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	1	1	0.11	0.11	0.11
UDP	1	1	1	1	0.11	0.11	0.11
MCS	1	1	1	1	0.11	0.11	0.11
SSL	1	1	1	1	0.11	0.11	0.11
SA	9	9	9	9	1	1	1
SV	9	9	9	9	1	1	1
STS	9	9	9	9	1	1	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.0323	0.0323	0.0323	0.0323	0.2903	0.2903	0.2903

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to End-to-End Security
Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	1	9	9	1	1
UDP	1	1	1	9	9	1	1
MCS	1	1	1	9	9	1	1
SSL	0.11	0.11	0.11	1	1	0.11	0.11
SA	0.11	0.11	0.11	1	1	0.11	0.11
SV	1	1	1	9	9	1	1
STS	1	1	1	9	9	1	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.1915	0.1915	0.1915	0.0213	0.0213	0.1915	0.1915

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to Certificates
Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	0.11	0.11	1	0.11	1
UDP	1	1	0.11	0.11	1	0.11	1
MCS	9	9	1	1	9	1	9
SSL	9	9	1	1	9	1	9
SA	1	1	0.11	0.11	1	0.11	1
SV	9	9	1	1	9	1	9
STS	1	1	0.11	0.11	1	0.11	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.0323	0.0323	0.2903	0.2903	0.0323	0.2903	0.0323

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to Users DB
Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	9	1	9	9	9
UDP	1	1	9	1	9	9	9
MCS	0.11	0.11	1	0.11	1	1	1
SSL	1	1	9	1	9	9	9
SA	0.11	0.11	1	0.11	1	1	1
SV	0.11	0.11	1	0.11	1	1	1
STS	0.11	0.11	1	0.11	1	1	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.2903	0.2903	0.0323	0.2903	0.0323	0.0323	0.0323

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to Security Service Token Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	1	1	1	1	0.11
UDP	1	1	1	1	1	1	0.11
MCS	1	1	1	1	1	1	0.11
SSL	1	1	1	1	1	1	0.11
SA	1	1	1	1	1	1	0.11
SV	1	1	1	1	1	1	0.11
STS	9	9	9	9	9	9	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.0667	0.0667	0.0667	0.0667	0.0667	0.0667	0.6

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to Flexibility Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	2	0.5	2	2	6
UDP	1	1	2	0.5	2	2	6
MCS	0.5	0.5	1	0.33	1	1	5
SSL	2	2	3	1	3	3	7
SA	0.5	0.5	1	0.33	1	1	5
SV	0.5	0.5	1	0.33	1	1	5
STS	0.17	0.17	0.2	0.14	0.2	0.2	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.1821	0.1821	0.1023	0.3019	0.1023	0.1023	0.0271

Consistency Ratio= 1.12%

Pairwise Comparison Matrix for the Alternatives with Respect to AVR RTT Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	0.99	1.18	0.76	0.78	1.21	6.8
UDP	1.01	1	1.19	0.76	0.79	1.22	6.85
MCS	0.85	0.84	1	0.64	0.66	1.02	5.75
SSL	1.32	1.31	1.57	1	1.03	1.6	9
SA	1.28	1.27	1.52	0.97	1	1.55	8.72
SV	0.83	0.82	0.98	0.63	0.65	1	5.64
STS	0.15	0.15	0.17	0.11	0.11	0.18	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.1554	0.1567	0.1313	0.2057	0.1993	0.1289	0.0229

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to STD RTT
Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	1	1	1	1	8.96
UDP	1	1	1	1	1	1	8.97
MCS	1	1	1	0.99	1	1	8.95
SSL	1	1	1.01	1	1	1	9
SA	1	1	1	1	1	1	8.99
SV	1	1	1	1	1	1	8.98
STS	0.11	0.11	0.11	0.11	0.11	0.11	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.1633	0.1635	0.1632	0.1641	0.1638	0.1638	0.0182

Consistency Ratio= 0.00%

Pairwise Comparison Matrix for the Alternatives with Respect to MAX RTT
Matrix

	UA	UDP	MCS	SSL	SA	SV	STS
UA	1	1	1	0.98	0.98	1	8.84
UDP	1	1	1	0.98	0.98	1	8.85
MCS	1	1	1	0.98	0.98	1	8.83
SSL	1.02	1.02	1.02	1	1	1.02	9
SA	1.02	1.02	1.02	1	1	1.02	8.99
SV	1	1	1	0.98	0.98	1	8.83
STS	0.11	0.11	0.11	0.11	0.11	0.11	1

Eigen Vector

UA	UDP	MCS	SSL	SA	SV	STS
0.1627	0.1628	0.1625	0.1656	0.1655	0.1624	0.0184

Consistency Ratio= 0.00%

Appendix C

A Generic AHP Tool

A Generic Java Class to Perform the AHP Calculations

```
public class AHP
{
    //The number of Criteria, Sub-Criteria and Alternatives

    public double numOfCriteria;
    public int[] numOfSubCriteria;
    public int numOfAlternatives;
    //Pair-wise Comparison Matrices
    public double[][] criteriaMatrix;
    public double[][][] subCriteriaMatrix;
    public double[][][][] alternativesMatrix;
    //Squared Matrices
    public double[][] criteriaSquaredMatrix;
    public double[][][] subCriteriaSquaredMatrix;
    public double[][][][] alternativesSquaredMatrix;
    //Normalized Matrices
    public double[][] criteriaNormalizedMatrix;
    public double[][][] subCriteriaNormalizedMatrix;
    public double[][][][] alternativesNormalizedMatrix;
    //Eigen Vectors
    public double[] criteriaEigenVector;
    public double[][] subCriteriaEigenVector;
    public double[][][] alternativesEigenVector;
    //Weighted Sum Vectors
    public double[] criteriaWeightedSum;
    public double[][] subCriteriaWeightedSum;
    public double[][][] alternativesWeightedSum;
    //Global Weights Matrices
    public double[][] globalWeight;
    //scoresVector Array
    public double[][][] scoresVector;
    public double[] results;
    //Consistancy Ratios
    public double criteriaConsistancyRatio;
    public double[] subCriteriaConsistancyRatio;
    public double[][] alternativesConsistancyRatio;
    //Saaty Random Consistancy Indices
    private static final double[] randomConsistancyIndices = new double[]
    {
        0,0,0,0.58,0.89,1.11,1.25,1.35,1.40,1.45,1.49
    };

    public AHP(int numOfCriteria, int[] numOfSubCriteria, int numOfAlternatives)
    {
        if (numOfCriteria == numOfSubCriteria.length)
        {
            //Initialize The numbers
            this.numOfCriteria = numOfCriteria;
            this.numOfSubCriteria = numOfSubCriteria;
            this.numOfAlternatives = numOfAlternatives;
            ///Initialize the Matrices
            criteriaMatrix = new double[numOfCriteria][numOfCriteria];
            criteriaSquaredMatrix = new double[numOfCriteria][numOfCriteria];
            criteriaNormalizedMatrix = new double[numOfCriteria][numOfCriteria];
            subCriteriaMatrix = new double[numOfCriteria][][];
            subCriteriaSquaredMatrix = new double[numOfCriteria][][];
            subCriteriaNormalizedMatrix = new double[numOfCriteria][][];
            for (int i = 0; i < subCriteriaMatrix.length; i++)
            {
                subCriteriaMatrix[i] = new double[numOfSubCriteria[i]][numOfSubCriteria[i]];
                subCriteriaSquaredMatrix[i] = new double[numOfSubCriteria[i]][numOfSubCriteria[i]];
                subCriteriaNormalizedMatrix[i] = new double[numOfSubCriteria[i]][numOfSubCriteria[i]];
            }
            alternativesMatrix = new double[criteriaMatrix.length][][][];
            alternativesSquaredMatrix = new double[criteriaSquaredMatrix.length][][][];
            alternativesNormalizedMatrix = new double[criteriaNormalizedMatrix.length][][][];
            for (int i = 0; i < alternativesMatrix.length; i++)
            {
                alternativesMatrix[i] = new double[subCriteriaMatrix[i].length][numOfAlternatives][numOfAlternatives];
                alternativesSquaredMatrix[i] = new double[subCriteriaSquaredMatrix[i].length][numOfAlternatives][numOfAlternatives];
                alternativesNormalizedMatrix[i] = new double[subCriteriaNormalizedMatrix[i].length][numOfAlternatives][numOfAlternatives];
            }

            //Initialise Vectors
            criteriaEigenVector = new double[criteriaMatrix.length];
            criteriaWeightedSum = new double[criteriaSquaredMatrix.length];
            subCriteriaEigenVector = new double[criteriaEigenVector.length][];
            subCriteriaWeightedSum = new double[criteriaWeightedSum.length][];
            for (int i = 0; i < subCriteriaEigenVector.length; i++)
            {
                subCriteriaEigenVector[i] = new double[numOfSubCriteria[i]];
                subCriteriaWeightedSum[i] = new double[numOfSubCriteria[i]];
            }
            alternativesEigenVector = new double[criteriaEigenVector.length][][];
            alternativesWeightedSum = new double[criteriaWeightedSum.length][][];
            scoresVector = new double[criteriaEigenVector.length][][];
            for (int i = 0; i < alternativesEigenVector.length; i++)
            {
                alternativesEigenVector[i] = new double[subCriteriaEigenVector[i].length][numOfAlternatives];
                alternativesWeightedSum[i] = new double[subCriteriaWeightedSum[i].length][numOfAlternatives];
                scoresVector[i] = new double[subCriteriaWeightedSum[i].length][numOfAlternatives];
            }

            //
            results = new double[numOfAlternatives];
            //Initialize Consistancy Ratios
            criteriaConsistancyRatio = 0;
            subCriteriaConsistancyRatio = new double[criteriaMatrix.length];
            alternativesConsistancyRatio = new double[criteriaMatrix.length][];
            for (int i = 0; i < alternativesMatrix.length; i++)
            {
                alternativesConsistancyRatio[i] = new double[subCriteriaMatrix[i].length];
            }
            //
            globalWeight = new double[numOfCriteria][];
            for (int i = 0; i < globalWeight.length; i++)
            {
                globalWeight[i] = new double[numOfSubCriteria[i]];
            }
        } else
        {
            System.out.println("numOfCriteria <> numOfSubCriteria.length");
            System.exit(0);
        }

    }

    public void squareMatrices()
    {
        //Calculate the Squared Matrices
        criteriaSquaredMatrix = squareMatrix(squareMatrix(criteriaMatrix));
        for (int i = 0; i < subCriteriaMatrix.length; i++)
        {
            subCriteriaSquaredMatrix[i] = squareMatrix(squareMatrix(subCriteriaMatrix[i]));
        }
        for (int i = 0; i < alternativesMatrix.length; i++)
        {
            for (int j = 0; j < alternativesMatrix[i].length; j++)
            {
                alternativesSquaredMatrix[i][j] = squareMatrix(squareMatrix(alternativesMatrix[i][j]));
            }
        }
    }

    private double[][] squareMatrix(double[][] array)
    {
        double[][] squaredMatrix = new double[array.length][array.length];
        for (int i = 0; i < array.length; i++)
        {
            for (int j = 0; j < array.length; j++)
            {
                for (int k = 0; k < array.length; k++)
                {
                    squaredMatrix[i][j] += array[i][k] * array[k][j];
                }
            }
        }
        return squaredMatrix;
    }

    public void normalizeMatrices()
    {
        //Calculate the Normalized Matrices
        criteriaNormalizedMatrix = normalizeMatrix(criteriaSquaredMatrix);
        for (int i = 0; i < subCriteriaSquaredMatrix.length; i++)
        {
            subCriteriaNormalizedMatrix[i] = normalizeMatrix(subCriteriaSquaredMatrix[i]);
        }
        for (int i = 0; i < alternativesSquaredMatrix.length; i++)
        {
            for (int j = 0; j < alternativesSquaredMatrix[i].length; j++)
            {
                alternativesNormalizedMatrix[i][j] = normalizeMatrix(alternativesSquaredMatrix[i][j]);
            }
        }
    }

    private double[][] normalizeMatrix(double[][] array)
    {
        double[][] normalizedMatrix = new double[array.length][array.length];
        double[] columnSum = new double[array.length];
        for (int j = 0; j < array.length; j++)
        {
            for (int i = 0; i < array.length; i++)
            {
                normalizedMatrix[i][j] = array[i][j];
                columnSum[j] += normalizedMatrix[i][j];
            }
        }

        for (int i = 0; i < array.length; i++)
        {
            for (int j = 0; j < array.length; j++)
            {
                normalizedMatrix[i][j] = normalizedMatrix[i][j] / columnSum[j];
            }
        }
        return normalizedMatrix;
    }

    public void calculateEigenVectors()
    {
        //Calculate Eigen Vectors
        criteriaEigenVector = calculateEigenVector(criteriaNormalizedMatrix);
        for (int i = 0; i < subCriteriaEigenVector.length; i++)
        {
            subCriteriaEigenVector[i] = calculateEigenVector(subCriteriaNormalizedMatrix[i]);
        }
        for (int i = 0; i < alternativesEigenVector.length; i++)
        {
            for (int j = 0; j < alternativesEigenVector[i].length; j++)
            {
                alternativesEigenVector[i][j] = calculateEigenVector(alternativesNormalizedMatrix[i][j]);
            }
        }
    }

    private double[] calculateEigenVector(double[][] array)
    {
        double[] eVector = new double[array.length];
        for (int i = 0; i < array.length; i++)
        {
            for (int j = 0; j < array[i].length; j++)
            {
                eVector[i] += array[i][j];
            }
            eVector[i] = eVector[i] / array.length;
        }
        return eVector;
    }

    public void calculateWeightedSums()
    {
        //Calculate Weighted Sums
        criteriaWeightedSum = calculateWeightedSum(criteriaMatrix, criteriaEigenVector);
        for (int i = 0; i < subCriteriaWeightedSum.length; i++)
        {
            subCriteriaWeightedSum[i] = calculateWeightedSum(subCriteriaMatrix[i], subCriteriaEigenVector[i]);
        }
        for (int i = 0; i < alternativesWeightedSum.length; i++)
        {
            for (int j = 0; j < alternativesWeightedSum[i].length; j++)
            {
                alternativesWeightedSum[i][j] = calculateWeightedSum(alternativesMatrix[i][j], alternativesEigenVector[i][j]);
            }
        }
    }

    private double[] calculateWeightedSum(double[][] matrix, double[] eigenVector)
    {
        double[] weightedSum = new double[eigenVector.length];
        for (int i = 0; i < matrix.length; i++)
        {
            for (int j = 0; j < matrix[i].length; j++)
            {
                weightedSum[i] += matrix[i][j] * eigenVector[j];
            }
        }
        return weightedSum;
    }

    public void calculateConsistencyRatios()
    {
        criteriaConsistancyRatio = calculateConsistancyRatio(criteriaWeightedSum, criteriaEigenVector);
        for (int i = 0; i < subCriteriaWeightedSum.length; i++)
        {
            subCriteriaConsistancyRatio[i] = calculateConsistancyRatio(subCriteriaWeightedSum[i], subCriteriaEigenVector[i]);
        }
        for (int i = 0; i < alternativesWeightedSum.length; i++)
        {
            for (int j = 0; j < alternativesWeightedSum[i].length; j++)
            {
                alternativesConsistancyRatio[i][j] = calculateConsistancyRatio(alternativesWeightedSum[i][j], alternativesEigenVector[i][j]);
            }
        }
    }

    private double calculateConsistancyRatio(double[] weightedSum, double[] eigenVector)
    {
        double eigenValue = 0;
        for (int i = 0; i < weightedSum.length; i++)
        {
            eigenValue += weightedSum[i] / eigenVector[i];
        }
        eigenValue = eigenValue / weightedSum.length;
        double ci = (eigenValue - (eigenVector.length)) / (eigenVector.length - 1);
        double ri = randomConsistancyIndices[eigenVector.length];
        double consistancyRatio = ci / ri;
        return consistancyRatio;
    }

    public void calculateGlobalWeight()
    {
        for (int i = 0; i < subCriteriaEigenVector.length; i++)
        {
            globalWeight[i] = new double[subCriteriaEigenVector[i].length];
            for (int j = 0; j < subCriteriaEigenVector[i].length; j++)
            {
                globalWeight[i][j] = criteriaEigenVector[i] * subCriteriaEigenVector[i][j];
            }
        }
    }

    public void calculateScores()
    {
        for (int i = 0; i < subCriteriaEigenVector.length; i++)
        {
            for (int j = 0; j < subCriteriaEigenVector[i].length; j++)
            {
                for (int k = 0; k < alternativesEigenVector[i][j].length; k++)
                {
                    scoresVector[i][j][k] = globalWeight[i][j] * alternativesEigenVector[i][j][k];
                }
            }
        }
    }

    public void calculateResultsVector()
    {
        for (int i = 0; i < subCriteriaEigenVector.length; i++)
        {
            for (int j = 0; j < subCriteriaEigenVector[i].length; j++)
            {
                for (int k = 0; k < alternativesEigenVector[i][j].length; k++)
                {
                    results[k] += scoresVector[i][j][k];
                }
            }
        }
    }
}
```

Java Main Class to Run the AHP Tool

```java
public class MainClass {

    /**
     * AHP Step 1: Create and Initialize Hierarchy
     */
    private int numOfAlternatives = 7;
    private int numOfCriteria = 3;
    private int[] numOfSubCriteria = new int[]{
        7, 4, 3
    };
    private final double[][] measuresArray = new double[][]{
        {
            6.798309352, 6.85455513, 5.746259492, 9, 8.719698985, 5.638189001, 1
        },
        {
            8.955703524, 8.969387467, 8.953079593, 9, 8.985187484, 8.981801766, 1
        },
        {
            8.838517131, 8.846820526, 8.828284664, 9, 8.992032095, 8.826131932, 1
        }
    };
    private String[] criteria;
    private String[] criteriaNames;
    private String[][] subCriteria;
    private String[][] subCriteriaNames;
    private String[][][] alternatives;
    private String[] alternativesNames;

    public MainClass() {
        //Initialise String Matrices
        criteria = new String[numOfCriteria];
        criteriaNames = new String[numOfCriteria];

        subCriteria = new String[criteria.length][];
        subCriteriaNames = new String[criteriaNames.length][];
        for (int i = 0; i < subCriteria.length; i++) {
            subCriteria[i] = new String[numOfSubCriteria[i]];
            subCriteriaNames[i] = new String[numOfSubCriteria[i]];
        }
        alternatives = new String[criteria.length][][];
        alternativesNames = new String[numOfAlternatives];
        for (int i = 0; i < alternatives.length; i++) {
            alternatives[i] = new String[subCriteria[i].length][numOfAlternatives];
        }

        //Initialize double Matrices
        AHP ahp = new AHP(numOfCriteria, numOfSubCriteria, numOfAlternatives);

        //Enter the names for all criteria and sub-criteria
        fillNamesArrays();


        /**
         * AHP Step 2: Pair-Wise Comparisons
         */
        //Enter the pair-wise compariosn values for the judgement-based criteria and sub-criteria
        fillStringArrays();

        //Calcualte the weights for the measurement-based sub-criteria
        for (int i = 0; i < measuresArray.length; i++) {
            weightsCalculator(ahp.alternativesMatrix[alternatives.length - 1][i], i);
        }

        /**
         * AHP Step 3: Eigen Vector
         */
        //Square all the matrices
        ahp.squareMatrices();
        //Normalize all the matrices
        ahp.normalizeMatrices();
        //calculate the eigen vector for each matrix
        ahp.calculateEigenVectors();
        // calculate the wighted sums
        ahp.calculateWeightedSums();

        /**
         * AHP Step 4: Consistency Ratio
         */
        ahp.calculateConsistencyRatios();

        /**
         * AHP Step 5: Ratings
         */
        ahp.calculateGlobalWeight();
        ahp.calculateScores();
        ahp.calculateResultsVector();

        //Display All the Matrices
        /*Code to display all the matrices and calculations*/
    }

    private void weightsCalculator(double[][] doubleArray, int index) {
        for (int i = 0; i < doubleArray.length; i++) {
            for (int j = 0; j < doubleArray[i].length; j++) {
                doubleArray[i][j] = measuresArray[index][i] / measuresArray[index][j];
            }
        }
    }

    private void fillStringArrays() {
        /**Code to enter the pair-wise comparisons*/
    }

    private void fillNamesArrays() {
        /**Code to enter names for the criteria and sub-criteria*/
    }

    public static void main(String[] args) {
        // TODO code application logic here
        MainClass m = new MainClass();
    }
}
```

Appendix D

An Example of a Stego Key

Sequence	Binary	Oct	Dec	Hex	Glyph
isbn-author-name-pages-publisher-	000 0000	0	0	0	NO EMBEDDING
isbn-author-name-publisher-pages-	010 0000	40	32	20	Space
isbn-author-pages-name-publisher-	010 0001	41	33	21	!
isbn-author-pages-publisher-name-	010 0010	42	34	22	
isbn-author-publisher-name-pages-	010 0011	43	35	23	#
isbn-author-publisher-pages-name-	010 0100	44	36	24	$
isbn-name-author-pages-publisher-	010 0101	45	37	25	%
isbn-name-author-publisher-pages-	010 0110	46	38	26	&
isbn-name-pages-author-publisher-	010 0111	47	39	27	'
isbn-name-pages-publisher-author-	010 1000	50	40	28	(
isbn-name-publisher-author-pages-	010 1001	51	41	29	)
isbn-name-publisher-pages-author-	010 1010	52	42	2A	*
isbn-pages-author-name-publisher-	010 1011	53	43	2B	+
isbn-pages-author-publisher-name-	010 1100	54	44	2C	,
isbn-pages-name-author-publisher-	010 1101	55	45	2D	-
isbn-pages-name-publisher-author-	010 1110	56	46	2E	.
isbn-pages-publisher-author-name-	010 1111	57	47	2F	/
isbn-pages-publisher-name-author-	011 0000	60	48	30	0
isbn-publisher-author-name-pages-	011 0001	61	49	31	1
isbn-publisher-author-pages-name-	011 0010	62	50	32	2
isbn-publisher-name-author-pages-	011 0011	63	51	33	3
isbn-publisher-name-pages-author-	011 0100	64	52	34	4
isbn-publisher-pages-author-name-	011 0101	65	53	35	5
isbn-publisher-pages-name-author-	011 0110	66	54	36	6
author-isbn-name-pages-publisher-	011 0111	67	55	37	7
author-isbn-name-publisher-pages-	011 1000	70	56	38	8
author-isbn-pages-name-publisher-	011 1001	71	57	39	9
author-isbn-pages-publisher-name-	011 1010	72	58	3A	:
author-isbn-publisher-name-pages-	011 1011	73	59	3B	;
author-isbn-publisher-pages-name-	011 1100	74	60	3C	<
author-name-isbn-pages-publisher-	011 1101	75	61	3D	=
author-name-isbn-publisher-pages-	011 1110	76	62	3E	>
author-name-pages-isbn-publisher-	011 1111	77	63	3F	?
author-name-pages-publisher-isbn-	100 0000	100	64	40	@

author-name-publisher-isbn-pages-	100 0001	101	65	41	A
author-name-publisher-pages-isbn-	100 0010	102	66	42	B
author-pages-isbn-name-publisher-	100 0011	103	67	43	C
author-pages-isbn-publisher-name-	100 0100	104	68	44	D
author-pages-name-isbn-publisher-	100 0101	105	69	45	E
author-pages-name-publisher-isbn-	100 0110	106	70	46	F
author-pages-publisher-isbn-name-	100 0111	107	71	47	G
author-pages-publisher-name-isbn-	100 1000	110	72	48	H
author-publisher-isbn-name-pages-	100 1001	111	73	49	I
author-publisher-isbn-pages-name-	100 1010	112	74	4A	J
author-publisher-name-isbn-pages-	100 1011	113	75	4B	K
author-publisher-name-pages-isbn-	100 1100	114	76	4C	L
author-publisher-pages-isbn-name-	100 1101	115	77	4D	M
author-publisher-pages-name-isbn-	100 1110	116	78	4E	N
name-isbn-author-pages-publisher-	100 1111	117	79	4F	O
name-isbn-author-publisher-pages-	101 0000	120	80	50	P
name-isbn-pages-author-publisher-	101 0001	121	81	51	Q
name-isbn-pages-publisher-author-	101 0010	122	82	52	R
name-isbn-publisher-author-pages-	101 0011	123	83	53	S
name-isbn-publisher-pages-author-	101 0100	124	84	54	T
name-author-isbn-pages-publisher-	101 0101	125	85	55	U
name-author-isbn-publisher-pages-	101 0110	126	86	56	V
name-author-pages-isbn-publisher-	101 0111	127	87	57	W
name-author-pages-publisher-isbn-	101 1000	130	88	58	X
name-author-publisher-isbn-pages-	101 1001	131	89	59	Y
name-author-publisher-pages-isbn-	101 1010	132	90	5A	Z
name-pages-isbn-author-publisher-	101 1011	133	91	5B	[
name-pages-isbn-publisher-author-	101 1100	134	92	5C	\
name-pages-author-isbn-publisher-	101 1101	135	93	5D	]
name-pages-author-publisher-isbn-	101 1110	136	94	5E	^
name-pages-publisher-isbn-author-	101 1111	137	95	5F	_
name-pages-publisher-author-isbn-	110 0000	140	96	60	`
name-publisher-isbn-author-pages-	110 0001	141	97	61	a
name-publisher-isbn-pages-author-	110 0010	142	98	62	b
name-publisher-author-isbn-pages-	110 0011	143	99	63	c
name-publisher-author-pages-isbn-	110 0100	144	100	64	d
name-publisher-pages-isbn-author-	110 0101	145	101	65	e
name-publisher-pages-author-isbn-	110 0110	146	102	66	f
pages-isbn-author-name-publisher-	110 0111	147	103	67	g
pages-isbn-author-publisher-name-	110 1000	150	104	68	h

pages-isbn-name-author-publisher-	110 1001	151	105	69	i
pages-isbn-name-publisher-author-	110 1010	152	106	6A	j
pages-isbn-publisher-author-name-	110 1011	153	107	6B	k
pages-isbn-publisher-name-author-	110 1100	154	108	6C	l
pages-author-isbn-name-publisher-	110 1101	155	109	6D	m
pages-author-isbn-publisher-name-	110 1110	156	110	6E	n
pages-author-name-isbn-publisher-	110 1111	157	111	6F	o
pages-author-name-publisher-isbn-	111 0000	160	112	70	p
pages-author-publisher-isbn-name-	111 0001	161	113	71	q
pages-author-publisher-name-isbn-	111 0010	162	114	72	r
pages-name-isbn-author-publisher-	111 0011	163	115	73	s
pages-name-isbn-publisher-author-	111 0100	164	116	74	t
pages-name-author-isbn-publisher-	111 0101	165	117	75	u
pages-name-author-publisher-isbn-	111 0110	166	118	76	v
pages-name-publisher-isbn-author-	111 0111	167	119	77	w
pages-name-publisher-author-isbn-	111 1000	170	120	78	x
pages-publisher-isbn-author-name-	111 1001	171	121	79	y
pages-publisher-isbn-name-author-	111 1010	172	122	7A	z
pages-publisher-author-isbn-name-	111 1011	173	123	7B	{
pages-publisher-author-name-isbn-	111 1100	174	124	7C	|
pages-publisher-name-isbn-author-	111 1101	175	125	7D	}
pages-publisher-name-author-isbn-	111 1110	176	126	7E	~
publisher-isbn-author-name-pages-	N/A	N/A	N/A	N/A	Control 1: To Continue
publisher-isbn-author-pages-name-	N/A	N/A	N/A	N/A	Control 2
publisher-isbn-name-author-pages-	N/A	N/A	N/A	N/A	Control 3
publisher-isbn-name-pages-author-	N/A	N/A	N/A	N/A	Control 4
publisher-isbn-pages-author-name-	N/A	N/A	N/A	N/A	Control 5
publisher-isbn-pages-name-author-	N/A	N/A	N/A	N/A	Control 6
publisher-author-isbn-name-pages-	N/A	N/A	N/A	N/A	Control 7
publisher-author-isbn-pages-name-	N/A	N/A	N/A	N/A	Control 8
publisher-author-name-isbn-pages-	N/A	N/A	N/A	N/A	Control 9
publisher-author-name-pages-isbn-	N/A	N/A	N/A	N/A	Control 10
publisher-author-pages-isbn-name-	N/A	N/A	N/A	N/A	Control 11
publisher-author-pages-name-isbn-	N/A	N/A	N/A	N/A	Control 12
publisher-name-isbn-author-pages-	N/A	N/A	N/A	N/A	Control 13
publisher-name-isbn-pages-author-	N/A	N/A	N/A	N/A	Control 14
publisher-name-author-isbn-pages-	N/A	N/A	N/A	N/A	Control 15
publisher-name-author-pages-isbn-	N/A	N/A	N/A	N/A	Control 16
publisher-name-pages-isbn-author-	N/A	N/A	N/A	N/A	Control 17
publisher-name-pages-author-isbn-	N/A	N/A	N/A	N/A	Control 18

publisher-pages-isbn-author-name-	N/A	N/A	N/A	N/A	Control 19
publisher-pages-isbn-name-author-	N/A	N/A	N/A	N/A	Control 20
publisher-pages-author-isbn-name-	N/A	N/A	N/A	N/A	Control 21
publisher-pages-author-name-isbn-	N/A	N/A	N/A	N/A	Control 22
publisher-pages-name-isbn-author-	N/A	N/A	N/A	N/A	Control 23
publisher-pages-name-author-isbn-	N/A	N/A	N/A	N/A	Control 24: End of Message