A Study on the Security Issues of Service Oriented Computing Paradigm in Network Environment

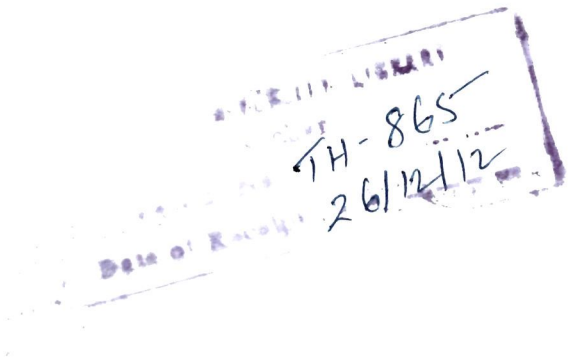
A Thesis Submitted to Assam University
In Partial Fulfillment of the Requirement for the Degree of

Doctor of Philosophy
In Computer Science

By

Subrata Sinha
Ph.D Registration No. 47100045 of 2005-06

DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF PHYSICAL SCIENCES
ASSAM UNIVERSITY
SILCHAR – 788011, INDIA
November 2010



DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF PHYSICAL SCIENCE
ASSAM UNIVERSITY, SILCHAR
(A CENTRAL UNIVERSITY CONSTITUTED UNDER ACT XIII of 1989)
SILCHAR -788011, ASSAM, INDIA

Date: 22/11/2010

CERTIFICATE

Certified that the thesis entitled "**A Study on the Security Issues of Service Oriented Computing Paradigm in Network Environment**", submitted by Subrata Sinha bearing Ph.D. Registration No. 47100045 of 2005-06 for award of the Degree of Doctor of Philosophy in Computer Science, is the outcome of a bonafide research work. This work has not been submitted previously for any other degree of this or any other university. It is further certified that the candidate has compiled this thesis fulfilling all the formalities as per the requirements of Assam University. We recommend that the thesis may be placed before the examiners for consideration of award of the degree of this university.

22/11/2010

Prof. Smriti Kumar Sinha
Joint Supervisor
Department of Computer Sc. & Engg.
Tezpur University, Napaam

22/11/2010

Dr. Bipul Syam Purkayastha
Supervisor
Department of Computer Sc.
Assam University, Silchar

DECLARATION

I, Sri Subrata Sinha, bearing Ph.D. Registration No. 47100045 of 2005-06 dated 18-05-2010, hereby declare that the subject matter of the thesis entitled "**A Study on the Security Issues of Service Oriented Computing Paradigm in Network Environment**" is the record of work done by me and that the contents of this thesis did not form the basis for award of any degree to me or to anybody else to the best of my knowledge. The thesis has not been submitted in any other University/Institute.

This thesis is being submitted to Assam University for the degree of Doctor of Philosophy in Computer Science.

Subrata Sinha
(Subrata Sinha)

Place: Silchar

Date: 22/11/2010

This thesis is dedicated

To

My parents

Sri Devdas Sinha and Srimati Sudevi Sinha

For

Showing me the light of this life

ACKNOWLEDGEMENTS

It gives immense pleasure to express my heartfelt gratitude to my supervisor Dr. Bipul Syam Purkayastha, Associate Professor, Department of Computer Science, Assam University, Silchar without his constant inspiration and encouragement; the present work would not have been possible.

I express my heartfull gratitude to my joint supervisor Prof. Smriti Kumar Sinha, Department of Computer Science and Engineering, Tezpur University, without his constant suggestion and guidance; the present work would not have been possible. When I approached him with the idea for a dissertation, he gave me the direction to approach the problem and taught me the research skill. Whenever I came to him with this problem he explained me the research methodology. During my stay in the department of Computer Science and Engineering in Tezpur University for the duration of one year, he extended all support and guidance and showed the direction at every chapter I used to complete. His valuable suggestions helped me to design my synopsis, abstract and finally the manuscript of this dissertation.

Special thanks to our honorable Vice Chancellor, Prof. Tapodhir Bhattacharjee of Assam University, Silchar to support me morally to complete my research work and relief me from my duty for the duration of one year to do the last phase of work under my joint supervisor in the Tezpur University, Napaam.

I would like to thank Prof. Debajyoti Biswas, Dean, School of Physical Sciences with whom I discussed my Ph.D related matters for a long time and who suggested me during my Ph.D thesis work.

Special thanks go to Mr. P. K. Dev Sharma, Head, Department of Computer Science, Assam University, Silchar for all kind of support during the course of my research work in

the department. Sincere thanks to all the faculty members, staffs and research scholars of the Computer Science department. Assam University, Silchar.

Special thank goes to Prof. K. Hemachandran, Department of Computer Science, Assam University, Silchar for his valuable advice during my research work. When I was in my tough period of my work he advised me to go for leave to complete the PhD thesis work.

My thank goes to Dr. Angshu Maan Sen, Director, Computer Centre Assam University Silchar and other staffs of the centre for moral support and taking all the pain of the centre during my absence. I would like to thank all my fellow colleagues and staffs of Assam University, Silchar who supported me directly or indirectly.

Thanks and gratitude is owned to many individuals for their help in various ways in the completion of this thesis. I would like to thank my parents Devdas Sinha and Sudevi Sinha, brother Sushanta Sinha, sisters Sheela and Supriya and others who supported me morally to complete my PhD thesis work. I would also like to thank my in laws Ramprasad Sinha and Late Sushama Sinha for their good wishes and full support. It gives me immense pleasure to acknowledge all those who have been the driving force in completing this PhD thesis work.

I thank my son, Sreejan, a gift from God, who always missed me during my absence. Last but not the least, I thank my wife Mrinalini Sinha for her tremendous support, care and encouragement throughout my PhD work. I am really grateful to her for taking all the pain during my absence.

Place: Silchar

Date: 22/11/2010

(Subrata Sinha)

Contents

List of Figures

List of Tables

Abstract

Service Oriented Computing (SOC) is becoming very popular and is being used to support many of the day-to-day workflows in large organisations. One of the major advantages with SOC is that it can be executed in heterogeneous and distributed environments. Since security is an essential and integral part of Web services, the SOC has to manage and execute the Web services in a secure way. In this thesis, different important concepts of secure Web services are investigated and presented.

The research work initiated with a study on the Security issues in Web services. In this thesis a study is done mainly on the security issues like integrity, digital signature, authorization and auditing in Web services. Firstly, a study is done on the limitations of Web service security on SOAP messages in providing end-to-end integrity in a document production workflow environment. The study has proposed a solution to overcome the limitations where a trusted central arbiter is used as an in-line TTP. Secondly, a study is done on the signature replacement attack where in a 2-tuple digital signature scheme if the signature element is replaced then there is no way to verify it. This attack is also equally true for XML signature scheme which is used in Web service security today. A solution has proposed with a BPEL process which acts as a central arbiter in the proposed special protocol. Thirdly, the issue of synchronization of authorization flow with a work object flow is presented and discussed. The study has shown how a work object flow is synchronised with the authorization flow using a central arbiter in Web services paradigm. The synchronization is achieved by exploiting the obligation provisions in XACML which provides authorization for Web services. Finally, a non-linear model called Extra-Tree is proposed to organize execution traces of Web services in a distributed computing environment in the form of a tree. The major advantage of this model is that the execution traces of Web services can be retrieved from the coarse-grained level to the fine-grained level of the tree as per the requirement.

Chapter 1

Introduction

This thesis contributes to the subject area of Web Services Security (WSS). It is mainly focused on the study and analysis of security techniques in Service Oriented Computing (SOC).

Today information is spread over many interconnected computers, termed as network. Internet is the world's largest network which contains almost every kind of information. With the availability of on-line resources anyone can search for many fields of knowledge deeply and thoroughly. The use of cryptography and encryption techniques also helps to address security issues in the Internet. The encryption technique [1] is a mathematical process involving the use of formulas or algorithms to protect the confidentiality, integrity and authentication of the information. It is the process of taking a message and scrambling it so that only the intended party can read it.

Services are simply a means for building distributed applications specifically how these applications are built and how services should function together harmoniously. The applications will use services by composing or putting them together. An architecture for service-based applications has three main parts: a provider, a requester and a registry. Providers publish or announce their services on registries, where requesters find and then invoke them. Services provide higher-level abstractions for organising applications for large scale open environments. Thus, they help us to implement and configure software applications in a manner that improves productivity and application quality.

Service-Oriented Computing (SOC) [2] is the latest paradigm of computing. It provides a way to create a new distributed architecture that reflects a trend towards autonomy and heterogeneity of the components. Service Oriented Computing (SOC) [2] can be viewed in terms of several different cross-cutting levels of abstractions ranging from those that concerns services within an application to those that concern service application interacting across enterprises.

Security is a major concern of Web services because of the fact that current trends in performing B2B (business-to-business) and B2C (business-to-consumer) transactions have been extended to the Web services. Web services facilitate different business processes belonging to various organizations into one single application.

This thesis aims to study the standard security issues in Web services.

1.1 Motivation

The growing trend of software architecture is to build platform-independent software component called Web services [3] that are available in the distributed environment of the Internet.

In Service Oriented Computing (SOC) [4], developers use services as fundamental elements in their application development processes. Services are platform and network independent operations that clients or other services invoke. To operate in an SOC environment, services must define their properties in a standard machine readable format. SOC offers three native capabilities – description, discovery and communication. For example, developers implement SOC native capabilities using Web Service Description Language (WSDL) for description, Universal Description Discovery Integration (UDDI) for discovery and integration and Simple Object Access Protocol (SOAP) for communication. There are four types of communications defined involving a service operation – (i) The end point resume a message, (ii) End point sends a message, (iii) End point receives a message and sends a correlated message and (iv) It sends a message and receives a correlated message. SOAP is an eXtensible Markup Language (XML) based means of passing messages that is intended to be language independent.

There are different hurdles to overcome if we look at the interoperation aspects in any system. The first is the connectivity among the applications, which protocols, such as HTTP, can readily ensure. The second is the ability of the various components to understand each other. Various enterprise policies must readily authenticate and authorize the parties involved in different interactions. A new problem arises when we want to introduce new applications and configure them to interoperate systems likely to be developed on different platforms and perhaps running on different operating systems.

Services can be implemented on a single machine, distributed on a LAN or even across several company networks. In all instances, a service must first be found and then it can be accessed. To this aim, each Service Oriented Architecture (SOA) relies on to distinct infrastructures called service discovery and service delivery. In most of the case, security is only addressed from the service delivery point of view and in some other cases it covers the discovery phase too.

This thesis represents a detailed analysis of the security requirements in Service-Oriented Computing (SOC) [5] which are still missing in the current literature. The ultimate goal of the security solutions is to provide security services such as authentication, confidentiality, availability and integrity to Web services. In order to achieve these goals, the security solution should provide complete protection spanning the entire protocol stack [6].

1.2 Objectives

Security is an important aspect of an information system; so far acceptability in the society is concerned. The main objectives of this proposed work is as follows:

1) To study the general security issues of Service Oriented Computing.
2) To study the special security issues of Service Oriented Computing.
3) To investigate the limitations of various security solutions and propose its counter-measures.
4) To develop the security techniques in Service Oriented Computing in different environments.
5) To provide solutions on various security protocols for evaluating better performance.

1.3 Methodology

In this thesis, the security issues and techniques of Service Oriented Computing are studied. The following steps are included as research work in this thesis:

1) A review on the literature on Service Oriented Computing (SOC) in general and security in specific was first done.
2) A study on the security mechanisms in SOAP messages was done and found the limitations of SOAP messages in Web Services Security. A solution was proposed using an in-line Trusted Third Party (TTP) to overcome these limitations.
3) A SWOT analysis on 2-tuple digital signature was then studied along with the findings of signature replacement attack. A solution was proposed to overcome this attack using an arbiter as a central server.
4) A technique was applied to synchronize the authorization flow with a work object flow using XACML and BPEL. The synchronization was done in architecture level and protocol level.
5) Finally, a model named Extra-Tree to organize execution traces of Web services was proposed.

1.4 Contributions

In this thesis, some security techniques in Web services are developed. Major parts of the works done in Web services security are focused on access control models. There are many more issues which are yet to be focused properly but important. In this thesis some such issues are addressed. In particular, there are four main contributions. These are -

1) To propose a solution on SOAP message integrity where Web Service Security (WSS) standard fails to ensure end-to-end security on SOAP messages.
2) To identify signature replacement attack and give a counter-measure.
3) To give a solution on the synchronization of authorization flow with a work object flow which has solved in architecture level and in protocol level.

4) To propose a model which organises execution traces of Web services using tree in distributed environment.

1.5 Thesis Organization

The rest of the thesis is organised as follows -

Chapter 2: Literature Survey on Web Services Security. This chapter presents an overview of Service Oriented Computing and its technologies. This chapter also provides a study on the security issues in Web services, the threats that they are facing today and the possible attacks that might be aimed at Web services.

Chapter 3: A Solution of SOAP Message Integrity. This chapter discusses on the limitations of Web Service Security in providing end-to-end integrity of a SOAP message in a document production workflow environment. A solution in BPEL process level using a special protocol is proposed.

Chapter 4: A Solution on Signature Replacement Attack. This chapter presents a signature replacement attack which is brought into the focus for the first time. The chapter proposes a solution for digital signature, resilient to signature replacement attack where a trusted central arbiter is used as an in-line TTP. The chapter proposes a solution with a BPEL process which acts as a central arbiter in the proposed special protocol.

Chapter 5: Synchronization of Authorization Flow with a Work Object Flow. This chapter presents the issue of synchronization of authorization flow with work object flow in a document production workflow environment. This chapter shows how a work object flow is synchronized with the authorization flow using a central arbiter. The outcome of this chapter is an authorization decision which may be either permit, deny, indeterminate or error occurred during evaluation.

Chapter 6: Extra-Tree: A Model to Organize Execution Traces of Web Services. This chapter presents a non-linear model, called Extra-Tree, to organize execution traces of Web services. This proposed model provides us a secured logging system which records the history of all the suspicious or malicious activities from the initiation of the Web service to

the completion of the Web service. The main focus of this chapter is to organize execution traces of Web services in a distributed environment in the form of a tree.

Chapter 7: Conclusions. Finally, this chapter presents the conclusions. Summary of the works and contributions are outlined along with the discussion of future research work.

Chapter 2

Literature Survey on Web Services Security

The evolution of Web services has facilitated the integration of business processes running in different platforms scattered across different geographical locations of the world. Along with the benefits that Web services provide for online transactions, it also poses some security threats. This chapter provides a study on the security issues in Web services, the threats that they are facing today and the possible attacks that might be aimed at Web services.

2.1 Introduction

Service Oriented Computing (SOC) [2] is a platform independent and language independent computing paradigm. SOC helps in implementing and configuring distributed software applications in a manner that provides productivity and quality with service orientation. Services are simply means for building distributed applications, where the objectives are mainly on how these applications are built and how services should function together. Service Oriented Architecture (SOA) is the standard architecture for this new computing paradigm. It defines an interaction between software agents as an exchange of messages among service requesters and service providers. Requesters are software agents that request for a service. Providers are software agents that provide services. The basic SOA includes, service provider, service requester and service registry. The interactions in SOA involves publish, find and bind operations. A service provider could be an industry, business or a company capable of providing services. The key requirement of any service provider is interoperability, security and performance [11]. A service requester also could be a company

or a business that is need of the service, where a service registry is a place, entity or a system that helps both service provider and service requester to discover each other.

Web service is the current technology of SOC. Web service technology implements SOA. The interface WSDL, registry UDDI and the message SOAP are built using XML and communicate via Internet protocols like HTTP, SMTP etc. XML is a platform independent language and a commonly accepted standard used for describing the different aspects of a Web service. XML and SOAP allow all kinds of systems to communicate with each other [27]. From the information exchange point of view, a Web service can be regarded as a simple mechanism to send and receive messages by a node [9]. Web service is a self contained, self describing, loosely coupled, re-usable software component that can be published, discovered and interacted via Internet protocols. Web services perform functions like transformation, storage and retrieval of data, aggregation, composition, orchestration etc. [10]. This chapter presents a latest review of research and development in Web services security.

2.2 Service Oriented Computing

Service Oriented Computing (SOC) [2] provides a better way to create a new architecture that trends towards autonomy and heterogeneity. Now-a-days, the current trend in the application space is moving from tightly-coupled systems towards loosely- coupled systems. The latest evolution in this new category of systems is a new paradigm, called Web services. The Web Service paradigm is the logical evolution from object oriented systems to the systems of services. As in object-oriented systems, some of the fundamental concepts in Web services are encapsulation, message passing and dynamic binding. Service Oriented Computing (SOC) is the new emerging paradigm for distributed computing and e-business processing that is changing the way software applications are designed, architected, delivered and consumed. Services are autonomous platform independent computational elements that can be described, published, discovered, orchestrated and programmed using standard protocols for the purpose of building agile networks of collaborating business applications distributed within and across organisational boundaries. Combined with recent developments in the area of distributed systems, workflow management systems, business protocols and languages, services can provide the automated support needed for e-business integration both at the data

and business logic level. They also provide a sound support framework for developing complex business transaction sequences and business collaboration applications. Adopting the SOC paradigm has the potential to bring about reduced programming complexity and costs, lower maintenance costs, faster time-to-market, new revenue streams and improved operational efficiency. However, before the SOC paradigm becomes a reality, there is a number of challenging issues that need to be addressed including among other things, like service modelling and design methodologies, architectural approaches, service development, deployment and composition, programming and evolution of services and their supporting technologies and infrastructure. The growing trend in software architecture is to build platform independent software components, called Web Services that are available in the distributed environment of Internet or Intranet. Architecture for service-based applications has three main parts: a service provider, a service requester and a service registry. Service providers publish their services on registries so that consumers can easily find the services and invoke them. A service requester can be a customer or a client or a person who need the particular service. The service registry process can be done through a basic protocol like UDDI (Universal Description Definition Integration).

2.3 Service Oriented Architecture

Service Oriented Architecture (SOA) is essentially a collection of services. These services are communicated with each other. The communication can involve either simple data passing or it could involve two or more services co-ordinating some activity. SOA allows a software programmer to model programming problems in terms of services offered by components to anyone, anywhere over the network. In other words, any application residing anywhere on any computer system would be able to interact with any service anywhere over the network. SOA defines an integration between software agents as an exchange of messages between service requesters (clients) and service providers. Clients are software agents that request the execution of a service. Providers are software agents that provide the services. Agents can be simultaneously both service requesters and service providers. Providers are responsible for publishing a description of the services that they provide. Clients must be able to find the descriptions of the services that they require and must be able to bind each other. The basic SOA is a relationship of three kinds of participants: the service provider, the service discovery agency and the service requester (client). The interactions involve publish, find and

bind operations. The service provider defines a service description of the service and publishes it to the service discovery agency through which a service description is published and made discoverable. The service requester uses a find operation to retrieve the service description typically from a discovery agency and then both the service provider and service requester negotiate each other by using bind operation.

In Web service models, *Service Provider* is the owner of the Web services. It holds the implementation of the service application and makes it accessible via the web. *Service Requester* represents a human or a software agent that intends to make use of some services to achieve a certain goal. *Service Registry* is a searchable registry providing service descriptions. It implements a set of mechanisms to facilitate service providers to publish their service descriptions. Meanwhile, it also enables service requester to locate services and get the binding information.

In Web services interacting modes, *Service Publishing* is to make the service descriptions available in the registry, so that the service requester can find it. *Service Finding* is to query the registry for a certain type of service and then retrieve the service description. *Service Binding* is to locate, contact and invoke the service based on the binding information in the service description.

2.4 Web Service

Web services are new concept that provides flexibility and connectivity between different systems. Web Services are using SOAP (Simple Object Access Protocol), a XML based protocol for communication over the network. A Web service is a software system identified by a URL, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by internet protocols. Web service is a loosely coupled software components that interact with one another dynamically via standard Internet technologies.

The Web services stack is as follows –

(1) *Communications*. A set of network protocols helps to realize the network accessibility of Web services such as HTTP, SMTP etc.

(2) *Messaging*. The messaging layer provides a document based messaging model for interaction with Web services.

(3) *Descriptions*. The description or representation layer is for describing Web services.

(4) *Discovery*. The discovery layer is for locating and publishing Web services.

(5) *Processes*. The processes layer supports more complex interactions between Web services, which enable Web service interoperation.

To compose Web services, the following things are required. These are –

1) Quality of Service (QoS)
2) Security
3) Reliability
4) Cost

This chapter identifies the main security requirements for Web services and describes how much security requirements are addressed by standards for Web services security recently developed or under development by various standardizations bodies.

2.5 Web Service Technologies

The technologies that form the foundations of Web services are SOAP, WSDL and UDDI. The international standard of SOAP is available in [12], where as those of WSDL and UDDI are available in [24] and [25] respectively.

2.5.1 SOAP

Simple Object Access Protocol (SOAP) is used for communication among different Web Services. SOAP [12] messages flow from originator to an ultimate receiver through a SOAP message path. A SOAP message consists of Soap: Envelope which contains a soap: Body element and an optional soap: Header element. The soap: Header element may contain a set of child elements.

```
01  <soap: Envelope --- →
02      <soap: Header (optional)>
03          <soap: Body> (mandatory)
04              <get Quote Symbol = "IBM"/>
05  </soap: Body>
06  </soap: Envelope>
```

Figure 2.5 (a): A Simple SOAP Message

Lines 01-06 contain the document root element named soap: Envelope. The syntax and semantics for the soap: Envelope are defined by SOAP. Line 02 contains soap: Header which is the optional part of the SOAP protocol. Soap: Header contains information for the SOAP node, the processor of the SOAP message, how to process the SOAP message. Lines 03-05 contain an element name soap: Body which is a child of the soap: Envelope element. Line 04 contains the get Quote element which is a child of the soap: Body. SOAP messages are nothing but XML documents.

2.5.2 WSDL

Web Service Description Language (WSDL) is an XML based language for describing functional properties of Web services. It aims at providing self-describing XML-based service definitions that applications as well as people can easily understand. In WSDL, a service consists of a collection of message exchange end-points. An end-point contains an abstract description of a service interface and implementation binding. The abstract description of a service contains – definition of messages that are consumed and generated by the service (i.e. input and output messages and – signatures of service operations. The implementation binding contains information about the location of a binding, the communication protocol to use (e.g. SOAP over HTTP) for exchanging messages with the service [10].

WSDL is written in XML. It is used to describe Web services. It is also used to locate Web services. A WSDL document is just a simple XML document. A WSDL document describes a Web service using the following elements [24] –

- *<type>*. The data types used by the Web services.
- *<message>*. The messages used by the Web services.
- *<portType>*. The operations performed by the Web service.
- *<binding>*. The communication protocols used by the Web services.

The Services and Ports define the location of Web services.

- *Port/Endpoint.* The Port contains the location of a Web service and the binding used for service access. The Port defines the connection point to a Web service.
- *Service*. The service contains the Web service name and a list of ports. It contains the collection of related endpoints. More details are available in [24].

```
<definitions>
        <types>
            definition of types ----------
        </types>
        <message>
            definition of message ----------
        </message>
        <portType>
            definition of a port ----------
        </portType>
        <binding>
            definition of binding ----------
        </binding>
</definitions>
```

Figure 2.5 (b): The Main Structure of a WSDL Document

2.5.3 UDDI

Universal Description Discovery and Integration (UDDI) is an XML-based registry for Web services. It is a place where businesses register and search for Web services. It is a directory for storing information about Web services. UDDI communicates via SOAP. UDDI uses WSDL to describe interfaces to Web services. A UDDI registry service is a Web service that manages information about service providers, service implementations and service metadata. Service providers can use UDDI to advertise the services they offer. Service consumer/requester can use UDDI to discover services according to their requirements and to obtain the service metadata needed to consume those services [25]. It defines an interface for advertising and discovering Web services. The UDDI information model identifies three types of information. These are –

- *White Pages.* White pages contain general information such as business name (i.e. a service provider's name) and contact information (i.e. service provider's phone no.).
- *Yellow Pages.* Yellow pages contain meta-data that can be used to effectively use to locate businesses and services based on classification schemes.
- *Green Pages.* The green pages contain service access information including service descriptions and binding templates. A binding template represents a service end-point (i.e. a service access interface) [10].

In UDDI registry, a Web service description contains business information, service information, binding information and information about specifications of services [27]. More details are available in [25].

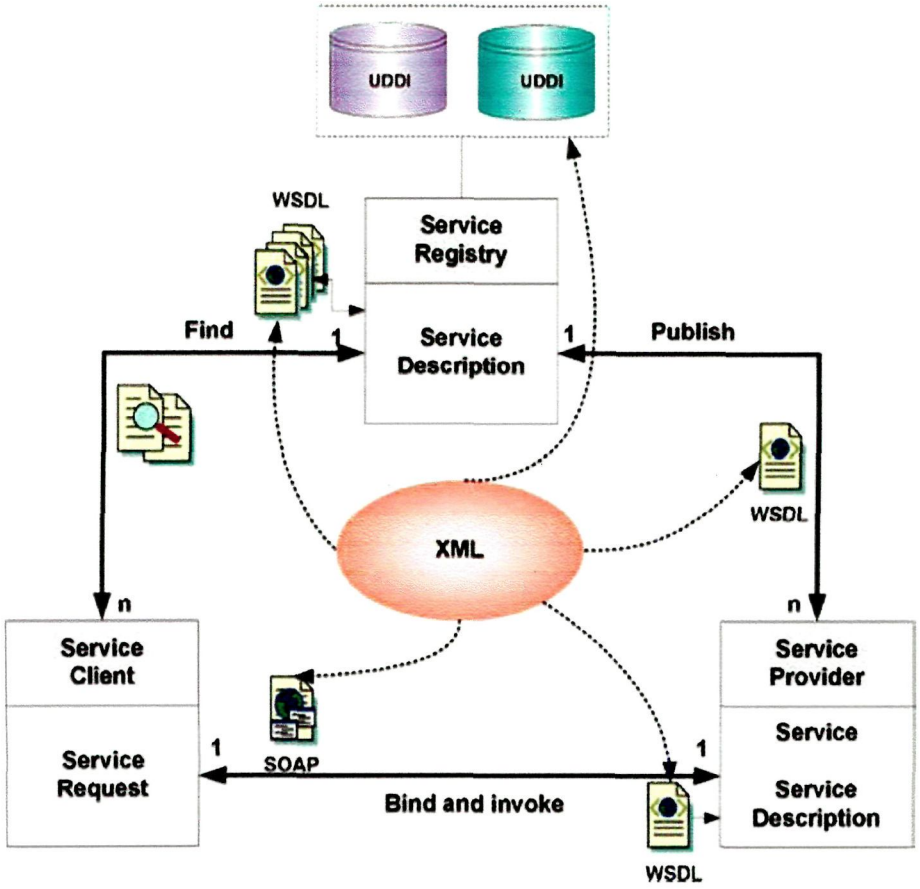


Figure 2.5 (c): Web Service Model

2.6 Security

The Web services security dimensions have been identified as, secure messaging, resource protection, negotiation of contracts, trust management and security properties. These dimensions encompass the security elements in a Web services environment which can be discussed broadly in the general security framework consisting of -

1) *Authentication*: Who is attempting to gain access. Authentication is concerned with the establishment of the proof of identities of entities in a system. The entity may be a user, a process or a service. Masquerading is a standard attack in authentication mechanism.

2) *Authorization*: Which resources is this requester allowed to access. Once an entity has been authenticated, the next issue is to ascertain which operations the entity is allowed

to do and on what resources. The authorization mechanism deals with granting and revoking privileges of authenticated entities.

3) *Confidentiality*: Can the data be read while in transit or storage. The issue of confidentiality specifies that only the sender and the intended recipient should be able to access the content of the message. An authorized person should not be able to access a message. It is achieved by encryption and decryption of messages. An encryption algorithm is used to convert plaintext into cipher text. There are two types of encryption in general use: symmetric and asymmetric encryption. In symmetric encryption, the decryption key is the same as the encryption key and in asymmetric encryption, the decryption key is not same as the encryption key. Eavesdropping is a standard attack in confidentiality.

4) *Non-repudiation*: Can a sender deny having sent a message. There are situations when a user sends a message and later on repudiates it. Repudiation may be on sending, receiving or on the time of sending or receiving the message as well.

5) *Availability*: Is this system vulnerable to Denial of Service (DoS) attack. The issue of availability states that resources, services should be available to authorized parties at all times. Denial of Service (DoS) is a standard attack on availability.

6) *Integrity*: Has the data or system been tempered with. When the content of a message is changed during transmissions than the integrity of the message is lost. Data integrity relies on mathematical algorithms known as hashing algorithms. A hashing algorithm takes a block of data as input and produces a much smaller piece of data as output. This output is sometimes called a digest of the data. If the data is a message, it is called a message digest. MD-5, SHA-1 and SHA-2 are the standard hashing algorithms.

7) *Auditing*: Is there a record of requester access to data. Security auditing is defined as involving the recognition, recording, storage and analysis of information related to security-relevant-activities. Audit records are intended to be examined to determine which security relevant activities took place and whom (which user) is responsible for them [28]. In computer security systems, audit trail is a sequential record of system resource usage. This includes user login, file access and also other various activities.

Under the above security elements, some special security issues are discussed as follows –

2.6.1 Authentication of Web Services

The requester, provider and registry in SOA need to be authenticated during composition, binding and execution of Web services. When a new service is composed by the service provider, the descriptions of that service are provided by the registry, and then the registry is to be authenticated by the requester. Web services may be vulnerable to man-in-the-middle attack, masquerading attack during composition, binding and execution. Various authentication techniques are discussed as follows:

1) *Single-Sign-On.* Once the end-user has been authenticated by its attributes login-id and password, it need not be authenticated again for communicating with the other services. Once the end-user sign on to a Web site and then a SOAP request is produced on the user's behalf, the route among multiple Web services is established using user's login-id and password. This functionality is known as Single-Sign-On (SSO) [15].

2) *Federated Trust.* Once the end-user has been authenticated using its attributes login-id and password, the route among multiple Web services may be established on the basis of their trust relationships. This mechanism is known as federated trust [16].

3) *WS-Trust.* To route among multiple Web services the trust relationships must be established among different Web services. The trust relationships among multiple Web services can be either direct or through the discovery agency.

4) *WS-Routing.* SOAP does not specify the actual message path along which a SOAP message is to travel. The message path consists of several intermediaries that a SOAP message will visit. SOAP needs to rely on the transport protocols and follow their message path models. Therefore, it is impossible to specify which intermediaries the SOAP message can visit when it travels to its destination. WS-routing enables SOAP path to be specified as a SOAP message header. It consists of a sequence of references to the intermediary SOAP nodes. When a message arrives at an intermediary node, the node will remove the reference thereof and send the message to the next node

based on the specified order. An intermediary node can also incorporate new nodes or remove existing nodes to change the message path [7].

5) *Multi-hoping.*

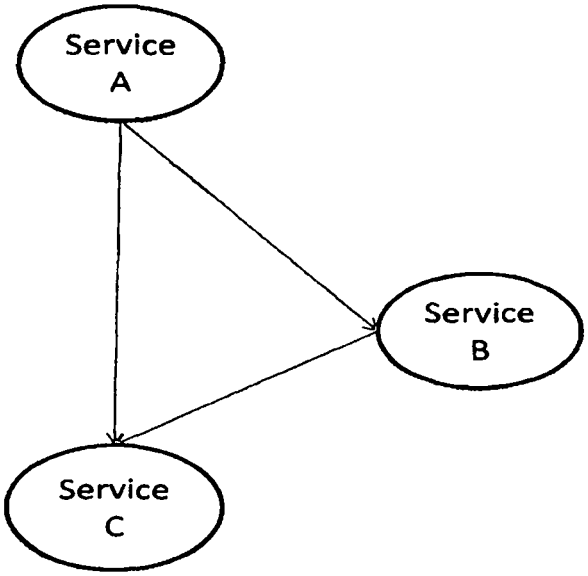


Figure 2.6 (a): An Example of Multi-Hoping

From the above figure it is assumed that,

$service_A$ authenticates $service_B$ => $service_A$ trusts $service_B$,

$service_B$ authenticates $service_C$ => $service_B$ trusts $service_C$,

Therefore, $service_A$ authenticates $service_C$ => $service_A$ trusts $service_C$.

The above example shows the case of multi-hoping, which means that the route through multiple Web services. This gives the concept of WS-routing where the SOAP messages are to route through the multiple Web services. The concept of WS-routing is not same with Single-Sign-On and Federated-trust, because the former considers only the SOAP messages has to route among multiple Web services. So far security challenges are concerned, mechanisms like Single-Sign-On and Federated-Trust may be designed on the top of WS-routing.

2.6.2 Authorization of Web Services

Once an entity has been authenticated, the next work is to ascertain which operations the entity is allowed to do and on what resources. The authorization mechanism deals with granting or revoking access of resources. The standard for authorization in Web service domain is XACML.

XACML. Data are more easily accessible today, by more people, so control of access is much more important. Due to this, there are regular requirements for the protection of data. The eXtensible Access Control Markup Language (XACML) is an XML based language which is required for expressing the rules needed to make authorization decisions. XACML includes access control policy language which defines the set of subjects that can perform particular actions on resources and the other one is request/response language, which is a way to express queries about whether a particular access should be "granted" or "revoked" [17].

The main objects that XACML deals with are attributes. Attributes are named values of known types. Specifically, attributes are the characteristics subject, resource, action or environment in which the access request is made [17]. Attributes are used in XACML to aid in creating access control policies. Attributes refer to individual properties of the subject, resource, action or environment that are applicable to the access request such as the subject's user name or environment's current time.

Access Control. Access control is generally defines as the prevention of unauthorized use of a resource which includes the prevention of use of a resource in an unauthorized manner. As confidentiality is a basic requirement for every interaction, access control is considered as an important security mechanism. Various authorization models are developed, the most notably known as Role Based Access Control (RBAC) and Attribute Based Access Control (CBAC) [18].

1) *RBAC*. RBAC (Role Based Access Control) is an authorization mechanism which is based on some roles that are applicable for authorized users. RBAC simplifies security management by providing a role hierarchy structure. Many organisation uses flexible platform independent XACML to support RBAC. RBAC on a Web service platform should be implemented for the administrator, developers and any other privileged accounts that will be required for the Web service to operate. The Web service platform

must be configured to enforce separation of roles that means not allowing a user assigned to one role to perform functions exclusively assigned to another role [16].

2) *ABAC.* ABAC (Attribute Based Access Control) provides a mechanism for representing either a user's or application's access profile through a combination of the following attribute types –

   a) *Subject Attributes (s)*: This is associated with a subject (user or application) that defines the identity and characteristics of that subject.

   b) *Resource Attributes (r)*: This is associated with a resource such as a Web service, system function or data.

   c) *Environment attributes (e)*: This attribute type describes the operational, technical and situational environment or context in which the information access occurs.

ABAC policy rules are generated as Boolean functions of s, r and e attributes and shows that whether a subject (s) can access a resource (r) in a particular environment (e) which is written as –

$$\text{Rule X : can\_access (s, r, e)} \leftarrow$$
$$\text{f(ATTR (s), ATTR (r), ATTR (e))}$$

ABAC generates with XACML, which relies on policy-defined attributes to make access control decisions [16].

***XACML Architecture* [23].** The XACML architecture specifies the implementation of a Policy Enforcement Point (PEP), a Policy Access Point (PAP), a Policy Decision Point (PDP), a Policy Information Point (PIP) and a Context Handler. Each of these are devoted to one specific task that is access control process. The PEP receives access requests from the requester and forwards them to the PDP which is responsible for the evaluation of attributes and which will take decision whether access is "granted" or "deny". The PIP supplies the attributes of subject, resource, action or environment to the PDP which are relevant for "allow" or "deny" access decision on resources. The access policy is provided by the PAP that stores and maintains the access rules. The XACML architecture also employs a Context Handler which manages a repository for all attribute definitions and their corresponding

implementations. The Context Handler is responsible for obtaining and supplying the requested values. The Context Handler shall return the values of attributes that match the attribute designator or attribute selector. If no attributes from the request context match then the attribute shall be considered missing. The authorization decision arrived at by the PDP is sent to the PEP with some obligations. The PEP fulfils the obligations based on the authorization decision sent by PDP which is either "permits" or "denies" access [19]. The data flow model of XACML architecture shown in Figure 2.6 (b) is as follows –

The PAP writes policies and policySets and makes them available to the PDP. The access requester sends a request for access resources to the PEP. The PEP sends the request for access to the Context Handler. The Context Handler converts the access request to an XACML request and sends it to the PDP. The PDP requests the attributes of subject, resource, action and environment from the Context Handler. The Context Handler requests attributes from the PIP. The PIP obtains the requested attributes and returns them to the Context Handler. The Context Handler sends the requested attributes and resource to the PDP. The PDP evaluates the policy. The PDP returns the response context including the authorization decision to the Context handler. The Context Handler returns the response to the PEP. The PEP fulfils the obligations. If access is permitted, then the PEP permits access to the resource; otherwise, it denies access [18], [19].

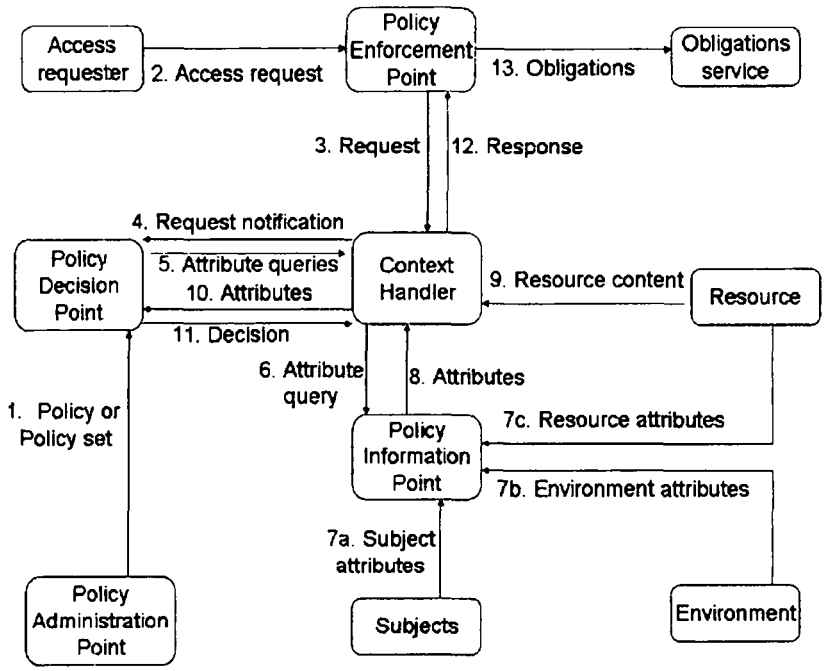


Figure 2.6 (b): Data Flow Diagram for XACML Architecture

2.6.3 Confidentiality of Web Services

XML encryption is mainly ensuring confidentiality to encrypt the XML data. XML encryption is the way to handle requirements for security in data interchange applications. XML encryption does not introduce any new cryptography algorithms or techniques for securing message. RSA, SHA-1, MD5 algorithms may still be used for actual encryption of message [20].

2.6.4 Non-repudiation of Web Services

WS-Security can provide non-repudiation of both the SOAP messages and its contents. WS-security supports signing the SOAP header, to ensure that the recipient and sender of the SOAP message have not changed since the message was sent [16]. Non-repudiation can be achieved by logging individual messages for later retrieval.

2.6.5 Availability of Web Services

The issue of availability states that, resources, services should be available to authorized users at all times. Availability is a major issue in Web service security. One of the means of denying availability is due to Denial-of-Service (DoS) attack. A DoS attack aims to use up all the resources of a service so that it is unavailable to users [20].

2.6.6 Integrity of Web Services

Integrity in Web services is mainly concerned with the WSDL files, which describes, defines or publish the functionalities of a Web service. When a service requester communicates with the registry for a suitable service, WSDL file is the basis of selecting the service during composition. XML signature is the technology that can be used for message integrity [16].

2.6.7 Audit Records and Mechanism

Records are necessary to enable a resolution if a party o a transaction denies the occurrence of the transaction or if other dispute arises; also to trace user access, behaviour and to enable system integrity verification [8].

2.6.8 Security Policy

A security policy [18] regulates the rules, requirements and mechanisms for Web service transactions in a large distributed environment. A security policy includes the following –

1) How the sender is authenticated, that is what mechanism is used with what parameters and in what value ranges.

2) Within the SOAP message, which XML elements are encrypted using what algorithms and key sizes and for what particular recipients or the recipient roles.

3) Within the SOAP message, which XML elements are integrity protected using what mechanisms with which algorithms and key sizes.

Enable implementers to define a security policy and enforce it across various platforms with varying privileges.

2.7 XML Security

XML documents are easily accessible which contributes lack of security in XML. Without knowing XML, one can decipher information easily and pick out sensitive information such as – name, address, credit card no. etc. This is the reason, why XML data is vulnerable. Two techniques broadly used to secure XML message are – XML encryption and XML digital signature [27].

2.7.1 XML Encryption

The XML encryption enables encryption of an entire XML document or specific parts of XML document. It is possible to encrypt a complete XML file or an element of XML file, non-XML data and the contents of an XML element. An element or contents of an element is the smallest part that can be encrypted. The only required element <CipherData> contains the encrypted content and stored in the XML document. The <EncryptionMethod> element is used to specify the encryption algorithm and the key size. The Key information provides how to decrypt cipher data and is stored in <KeyInfo> element [27]. XML Encryption allows XML syntax to be encrypted to ensure data confidentiality. The encrypted syntax is replaced by an <EncryptedData> element containing the ciphertext of the encrypted data as content. The XML encryption also defines an <EncryptedKey> element for any transportation purposes. An XML data is encrypted with a randomly generated symmetric key, which itself is encrypted using the public key of the message recipient. In SOAP messages, the <EncryptedKey> element – if – present – must appear inside the security header [20].

2.7.2 XML Digital Signature

XML digital signature allows XML syntax to be digitally signed to ensure integrity or to proof of authenticity [20]. The XML digital signature standard defines a schema discussed in Figure 2.7 for capturing the result of a digital signature operation applied to XML data or other data types. XML signatures add authentication, data integrity and non-repudiation to the data that they sign. A fundamental feature of XML signature [14] is the ability to sign only specific portions of the XML tree rather than the complete document. XML signature is an evolving standard for digital signatures [22].

The XML digital signature schema discussed in Figure 2.7 is a 2-tuple signature, where the signed element is referenced by its URI under the <Reference URI?> element [26]. XML signature schema uses URI to identify resources, algorithms and semantics. The <SignedInfo> element contains the signed data and specifies what algorithms are to be used. The <SignatureMethod> and <CanonicalizationMethod> are used by the <SignatureValue> element and are included in <SignedInfo> to protect them from tempering. The <SignatureMethod> is the algorithm that is used to convert the canonicalzed <SignedInfo>

into the <SignatureValue>. It is a combination of a digest algorithm and a key dependent algorithm and possibly other algorithms, for example RSA, SHA-1, MD-5 etc. The <CanonicalizationMethod> is the algorithm that is used to canonicalize the <SignedInfo> element. Each <Reference> element includes the <DigestMethod> and the resulting <DigestValue> calculated over the identified (by URI) data object. It also may include <Transforms> that produce the input to the digest operation. One or more <Reference> elements specify the resource being signed by URI reference and any <Transforms> to be applied to the resource prior to signing. <DigestMethod> specifies the Secure Hash Algorithm (SHA-1) before applying the one-way hash function. <DigestValue> contains the result of applying the hash algorithm to the transformed resources. The <SignatureValue> element contains the digital signature value that is an encrypted digest of the SignedInfo. The signature generated with the parameters specified in the <SignatureMethod> element of the <SignedInfo> element after applying the algorithm specified by the <CanonicalizationMethod>. <KeyInfo> (optional) indicates the key to be used to validate the signature. Since, <KeyInfo> is outside of <SignedInfo>, if the signer wishes to bind the keying information to the signature, a <Reference> can easily identify and include the <KeyInfo> as part of the signature. The <Object> element (optional) contains any other information to support the signature [22].

```
<Signature ID?>
   <SignedInfo>
     <CanonicalizationMethod/
     <SignatureMethod/>
     (<Reference URI? >
       (<Transforms>)?
       <DigestMethod>
       <DigestValue>
     </Reference>)+
   </SignedInfo>
   <SignatureValue>
  (<KeyInfo>)?
  (<Object ID?>)*
 </Signature>
```

Figure 2.7 [13]: The Structure of XML Digital Signature

From the above discussion, it can be noted that, XML digital signature and XML encryption solves many problems, but they do not solve all the problems. XML signature involves many algorithms and the strength of the signature depends on the used digest algorithms [27].

2.8 Secure Socket Layer

SSL (Secure Socket Layer) is a technology that is used to protect companies from Web Service Security (WSS) attacks. SSL used in encryption technique, which are in turn used to implement for data protection. SSL creates a secure tunnel in between source and destination computers based on public key encryption technique. A common protective measure is to send messages over a secure connection that is using SSL. For example, an SSL connection between two computers may be sufficient for simple applications. For multiple Web services, complete message or individual part of messages may be encrypted and signed to protect the confidentiality and integrity of Web service messages [21].

It is therefore very important to have a tool to guarantee that the message was not modified in transit and that each party can be sure that no other fraudulent party is participating. One of the first counter-measures is the usage of SSL.

1) SSL allows the client to determine that the service is the real service that the client is supposed to communicate with. The SSL server (i.e. the service provider) sends his certificate, which is checked by the client (application).

2) SSL guarantees message integrity. All traffic between the two TCP endpoints is protected from forgery.

3) SSL encrypts all traffic, so that the (even plaintext XML) messages cannot read by unauthorized parties.

4) SSL is widely available and understood by all major browsers. Also, SSL can easily be configured for Web services and commercial applications usually support it as well.

5) SSL has one disadvantage, which is that encryption and other security functions are performed only by the two TCP endpoints. If the message needs to be secured further,

then an end-to-end protection like the usage of XML signature and XML encryption in the application layer is required.

2.9 WS-Security

WS-SecureConversation	WS-Federation	WS-Authorization
WS-Security Policy	WS-Trust	WS-Privacy
WS-Security : SOAP Message Security		

Table 2.9: The WS-Security Family

The WS-Security related specifications define the required policies to properly secure the WS interactions. The task of WS-Security is to set the constraints and capabilities of a WS and they do not have intension to substitute any security technologies. On the contrary, WS-Security related specifications enlarge and merge existing security infrastructure and correctly define how these can be used in an interoperable way [11]. WS-Security is achieved by defining header elements to be included in SOAP messages. WS-Security does not provide a complete security framework for Web services. However, it provides mechanism for ensuring single-message security within SOAP. In WS-Security, message integrity of SOAP messages provided using the XML signature specification in conjunction with security tokens. In WS-Security, message confidentiality uses the XML encryption specification in conjunction with security tokens [10]. The key aspects of WS-Security layers are as follows –

1) *WS-Security/SOAP Message Security.* It is made on the SOAP specification and specifies how to sign and secure SOAP messages.

2) *WS-Trust.* To route among multiple Web services, the trust relationship must be established among different Web services. The trust relationships among multiple Web services can be either direct or through the discovery agency [16].

3) *WS-SecureConversation.* It is built based on WS-security and WS-trust to specify how WS can manually manage and authenticate security contexts. It includes describing how service requesters can authenticate WS, as well as how WS can authenticate messages from service requesters.

4) *WS-Security Policy.* It specifies a generic format through which to describe security capabilities and requirements for SOAP message senders and receivers.

5) *WS-Federation.* It is made on all previous specifications to specify how to broker and manage heterogeneous, federated trust contexts.

6) *WS-Privacy.* It is built based on WS-Security, WS-Security Policy and WS-Trust to specify a model by which organizations using WS can indicate preferences as well as conformance to particular privacy policies.

7) *WS-Authorization.* It specifies how to access policies for WS are specified and managed using a flexible authorization language (XACML) and format [11].

2.10 Threats and Attacks

Security decisions must always be made with an understanding of the threats facing the system to be secured. The possibility of an attack is known as threat. The common threats of Web services are as follows [8] –

1) *Message alteration.* In this threat, message information is altered by inserting, removing or modifying information created by the originator of the information.

2) *Loss of Confidentiality.* This type of threat makes information within the message visible to unauthorized participants.

3) *Falsified messages.* This type of threat occurs when an attacker constructs new messages and sends them to a receiver who believes that the message has originated from a third party other than the sender.

4) *Principal spoofing.* A message is sent which appears to be from another principal. For example, Alice sends a message which appears as if it is from Bob.

5) *Forged claims.* A message is sent in which the security claims are forged in an effort to gain access to unauthorized information. For example, a security token which was not exactly issued by the specified authority.

6) *Replay of message parts.* A message is sent which includes portions of another message in an effort to gain access to unauthorized information.

7) *Replay of message.* A whole message is resent by an attacker.

The typical requirements for a secure system are integrity, confidentiality and availability. Any action targeting at the violation of one of these Web service security properties is called an attack. The possibility of an attack is known as vulnerability. Web services may be vulnerable to man-in-the-middle attack, masquerading attack, eavesdropping attack, Denial-of –Service (DoS) attack etc. during composition, binding and execution. Some other attacks on Web service applications like SQL injection or parameter tempering which also use non-valid messages focusing on integrity. The list of attacks that facing by Web services are [27]
-

1) *Buffer Overflow Attack.* This type of attack arises when the amount of reserved for the operation becomes smaller than the amount of data written to the memory. Poorly written code, such as a program that inserts data into a buffer and does not check the size of the data being inserted, often invites buffer overflow attacks.

2) *Cross Site Scripting (XSS or CSS).* In this attack, the attacker inserts malicious code in the request and this will be returned to the victim by that application. While the script runs the attacker can perform the attack on behalf of the user. The majority of XSS or CSS attacks rely on <scripts> tags. An XSS attack sets up, for example, a Trojan horse in the victim's Web browser. The Trojan horse is often created by client-side languages such as JavaScript, Java, and VBScript or takes advantage of a known vulnerability in the browser.

3) *IP–Spoofing.* Performing IP-Spoofing attack, an attacker fakes IP address to deceive receiver to believe it is sent from a location that it is not actually from. If an attacker gains access to the network with a valid IP-address, he/she can modify, reroute or delete data.

4) *SQL Injection.* The SQL injection attempts to manipulate the application's database by using raw SQL statements. The attacker can execute arbitrary commands in the database using this attack.

5) *Network Eavesdropping.* This attack occurs when an attacker gained access (possibly through IP-Spoofing or Man-in the-Middle attacks) to the data path to the specific network. To performing this attack, the attacker can capture traffic and obtain usernames and passwords.

6) *Dictionary Attack.* An attacker systematically tests all possible passwords to perform dictionary attack. Here, the attacker's goal is to obtain passwords. To perform dictionary attack, the attacker trying every word in the dictionary until matching password is found. Most password-based authentication algorithms are vulnerable to dictionary attack.

7) *Data Tempering.* Data tempering attack occurs when an attacker changes or modifies valid data, while it passes over the network. Any data sent from the client-side may be manipulated by an attacker. Successful attacks results in tempered data that reaches to the receiver.

8) *Denial-of-Service (DoS) Attack.* In DoS, the attacker's goal is to disclose information that he can use for crashing the Web application process. Performing DoS, the attacker may attack a router, firewall or proxy server with the goal of making them unusable. The effect of DoS attack is to stop the service-providing computer from being able to provide the service.

9) *Man-in-the-Middle Attack.* When an attacker intercepts messages between client and server, the man-in-the-middle attack can occur. Both client and server assume that they are communicating each other, but the communication between the sender and the receiver is actually flowing through the attacker. The attacker is free to modify the content of the messages and sent them to the receiver. The receiver receives the message that she thinks that the message has come from the sender and acts on it. Then the receiver sends the message back to the sender, which goes through the

attacker. Unfortunately, neither sender nor receiver knows that they have been attacked.

10) *Parameter Tempering.* The aim of parameter tempering attacks is to modify parameters sent between the user and the application. An attacker may change the URI parameters or mode parameter. The parameter tempering includes Query string manipulation, Form field manipulation and HTTP header manipulation.

More details of these attacks on Web services are available in [27]. This chapter have not provided any solution on how to mitigate these attacks but they have focused on the source of generating possible attacks on SOAP message implementation of XML Web services.

Some of the new attacks are summarized below which are belongs to the category of DoS attacks focusing on availability [20]. These attacks are -

1) *Oversize Payload.* In Web services, the oversize payload attack is caused due to the high memory consumption of XML processing. The total memory usage caused by processing one SOAP message is much higher than the message size. This is due to the fact that, most Web services implement a tree-based XML processing model.

2) *XML Injection.* An XML injection attack tries to modify the XML structure of a SOAP message or any other XML document by inserting content – e.g. operation parameters containing XML tags. Such attacks are possible if the special characters "<" and ">" are not escaped properly.

3) *SOAPAction Spoofing.* The optional HTTP header field "SOAPAction" can be used for operation identification. The SOAPAction field value is often used as the only qualifier for the requested operation. The SOAPAction spoofing attack is happened when the server actually ignored the SOAPAction value and invoked another operation when a message is sent to the service.

4) *WSDL Scanning.* If the Web service is created using common Web service framework tools, the generated WSDL contains all operations. In this case, an external client gains knowledge of the internal operations and can invoke them. For example, an Internet shop system needs methods for placing an order for customers (sendOrder) and for administrating the orders (adminOrders). If both sendOrder and adminOrder

operations are offered by one Web service, an attacker with the knowledge of sendOrder can easily find the administration method also.

5) *Metadata Spoofing.* A Web service client retrieves all information regarding a Web service invocation (i.e. message format, network location, security requirements etc.) from the metadata documents provided by the Web service server. Currently, this metadata usually is distributed using communication protocols like HTTP or mail. These circumstances open new attack possibilities aiming at spoofing these metadata. The most relevant attacks in this category are WSDL spoofing and security policy spoofing. More details of these attacks are available in [20].

2.11 Chapter Summary

Web service has emerged today as an enabling technology for business transactions across different platforms and languages where business processes are encoded and running. It is the technology which enables distributed computing mainly in the form of workflows. Security is a major concern for social acceptance over and above the platform independent and the language independent of the technology. This chapter mainly reviews the security aspects of Web services. It is seen that the security concerns are similar to other web applications in general; however Web services have specific security issues as well. The specific security challenges are mainly due to the adoption of XML in different components of Web services. The present study includes recent threats and attacks specific to XML-based Web services.

Chapter 3

A Solution on SOAP Message Integrity

SOAP messages with XML signatures under Web Service Security specification provide secure message exchange solutions in SOA based applications. Recent researches established that the solutions based on the specification have several limitations. XML rewriting attacks on SOAP messages exposed the vulnerability of SOAP messages and different solutions are proposed to counter the attacks. This chapter exposes few other limitations of Web service security in providing end-to-end integrity, specially part integrity and reuse issues, of multiple signed messages in a SOAP message in a document production workflow environment. This chapter also discusses the consequences of the limitation and establishes that it is not possible to address these issues at message level. It also proposes a solution in BPEL process level using a special protocol.

3.1 Introduction

Business processes today are modelled as workflows. A workflow is defined as a set of coordinated tasks. Recent trend in software industry is towards adopting Service Oriented Architecture (SOA) in applications. SOA applications are flexible, dynamically composable and inter operable. Web services provide a technology that can be used to implement SOA and are increasingly becoming the SOA implementation of choice. In a business workflow, tasks are implemented as Web services. Functionalities of each Web service is described in an interface, which is stored in a separate file. The interface is described using a language called Web Service Definition language (WSDL). Since the WSDL interface and the actual

implementation of the service are decoupled, the implementation can be replaced by another implementation compliant to the published interface [24]. The Web services are co-ordinated by another process, which is specified by Business Process Execution Language (BPEL). The type of coordination done by the BPEL process is known as *orchestration*. The interface of the BPEL process is also defined by another WSDL file. Therefore, a BPEL process is a composite Web service, which in turn may be used in the composition of other composite Web services [29]. Among Web Services, communications are made through Simple Object Access Protocol (SOAP) messages. SOAP [12] is a W3C specification used for exchanging messages among Web services. WSDL, BPEL, SOAP are all derivatives of XML languages. Therefore, the foundation of Web service technology is on XML. An XML document is a tree having a root element, which may have multiple elements as children, each child has further child elements and so on. Each element may have multiple attributes. All elements and attributes are user defined, no predefined elements and attributes. It's a specification to define other markup languages. SOAP messaging framework consists of a message construct, a processing model, an extensibility model and protocol binding to multiple underlying transport protocols, like HTTP, FTP, SMTP etc. According to message construct the root element of a SOAP message is *envelope*, which has an optional *header* and a mandatory *body* element. The *header* element contains the meta-data only, under multiple header blocks, no application payload. The *body* has the actual message, as XML payload. SOAP specifies a distributed processing model which allows multiple intermediaries to process the message and thus insert multiple new messages during the itinerary of the original message from the original sender to the final receiver. This facilitates the use of SOAP framework in a workflow environment. Security is an important requirement for any practical business workflow. To provide message level security during storage and communication among multiple Web services, OASIS published a standard specification on Web Service Security: SOAP Message Security 1.1(WSS) [30], [31].WSS uses XML signature [22] as a security mechanism to address proof of origin and content integrity issues of XML messages. It also allows to sign not only the whole XML payload of a SOAP message but also a part of it. The goal of WSS is to ensure end-to-end message level security. When a message that traverses multiple intermediaries, within and between business entities e.g. companies, divisions and business units, it is secure over its full route through and between those business entities. This includes not only messages that are initiated within the entity but also those messages that originate outside the entity, whether they are Web Services or the more traditional messages [30].

XML security is the basis for Web Services Security (WSS). In one direction, for access control, we have XACML specification. Research works on XACML in different environments are going on. For example, XACML as a security pattern and integration scheme, suitable for workflows in ambient intelligence environment is discussed in [32]. In the other direction, for message level security, works are going on towards secure SOAP messages and WSS provides the required security. WSS specification is flexible, but unfortunately, SOAP messages compliant to this specification are still vulnerable. A SOAP message is prone to attacks that can lead to several consequences, such as, unauthorized access, disclosure of information, identity theft etc. In the literature, the attacks are basically on-the-fly modifications of SOAP messages and are referred to as XML rewriting attacks [33]. Providing a solution to XML rewriting attack on SOAP messages is a vital challenge of Web Service Security. The attacks are framed on the weaknesses of XML signature. This opens up the flush door to security attacks and makes secure SOAP message exchange vulnerable [26]. One of the main security research focuses is XML rewriting attack. Different solutions for XML rewriting attacks are proposed in the literature to address this problem [13], [34], [35]. Recently, the first formal and almost complete solution to the XML rewriting attack was proposed [26]. The study motivated us to explore further limitations of WSS specification specially in a document production workflow in an office environment.

Apart from weaknesses of XML signature in SOAP messages, which led to XML rewriting attacks, it is found from the investigation that there are few more limitations of current WSS standard on SOAP messages, hitherto not discussed in Web Services Security literature but equally important, specially in the cases of workflow implementations as orchestrated or choreographed composite Web services. In this chapter, these limitations are brought to focus. Current version of WSS fails to address these issues. These were first discussed in [36] as issues of Multi-Part Multi-Signature Document (MPMSD) in the context of Document Production Workflow (DPW) in an office and a solution were proposed in [37]. These issues are found to be still relevant and important in SOAP security. In WSS standard, XML signatures and XML encryption [22] with inclusive and exclusive canonicalization ensure proof of origin, content integrity and confidentiality issues of each individual part, that is, each signed message, in a SOAP envelope during exchange and storage. But it fails to ensure the collective integrity issue of all the parts as a whole, which means all the signed messages in a SOAP envelope signed by the original sender and the intermediaries during the workflow. It also fails to address the part reuse issue [37]. WSS standard fails to ensure end-

to-end security in SOAP message level on these issues. As a consequence several attacks may be possible, similar to the vulnerabilities described in the context of MPMSD [37]. It is also shown in this chapter that it is not possible to ensure resolution of these issues in the SOAP envelope directly. It has to be ensured in the BPEL process level through a protocol.

3.2 MPMSD

Document production in an office is based on a *request-reaction-response* paradigm. When a message containing a request is received in an office, the office reacts to the request. The reactions are recorded in the form of comments on the document and finally a response message is dispatched. We can term the process as Document Production Workflow (DPW). The resultant document of a DPW is termed as a Multi-Part Multi-Signature Document (MPMSD). Therefore, a MPMSD is a case of the DPW. The first part of a MPMSD is the request message and the last part is the response message and the intermediate parts are the messages containing the comments of other reviewers, which means, the reactions. Each part of a MPMSD is signed by the corresponding reviewer. The first reviewer is also termed as the originator. MPMSD is a generic framework. Parts belonging to different cases may also form a MPMSD. Moreover, multiple versions of a document may also be defined as a MPMSD. A reviewer creates a part of a MPMSD in the context of a set of existing documents constituted of rules, precedents and other support documents, which in turn may be MPMSDs. Rules, precedents and support documents constitute a reference space of an office. Here, in this chapter it has discussed only it from the work object or case point of view in a document production workflow [38].

A Multi-Part Multi-Signature Document (MPMSD), D_W, produced in a DPW W, is an n-tuple, $n \geq 1$, such that $D_W = (d_1, d_2, d_3, \cdot \cdot \cdot, d_n)$. Each part d_i in turn is defined as a 3-tuple (m_i, σ_i, s_i), where m_i is the comment of the reviewer s_i, and σ_i is the signature of s_i. MPMSD is basically an abstract representation of a SOAP envelope, containing multiple signed messages.

3.2.1 A Scenario

To understand the salient features of a DPW in an office let us consider the following scenario of a DPW. An employee, A submits an application, m_A regarding her travel plans for approval to the manager, B of the department. B verifies the travel plans in the context of previous cases of employees from the team already in travel, type of leave to be granted for A during travel, standing rules, etc. and adds her comment, m_B and forwards it to the finance officer, C. C also examines the case by verifying the budget allocation status under the head of account for travel, TA/DA rules in such cases and adds her comment, m_C on the amount that may be granted and forwards it to the director, D. D also justifies the previous comments, approves the travel plans and adds the note of approval, may be in the form of office order, m_D. A copy of the whole multipart document or only the office order m_D may finally go back to the originator A and the original multi-part document is stored in a folder. The flow of the document is recorded in log-books. This is a case of the travel plan workflow. It is shown in figure 3.2.

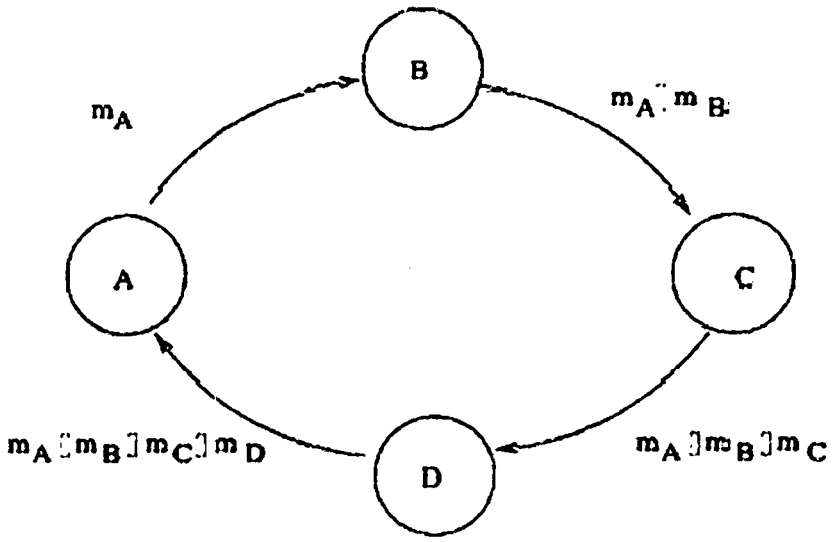


Figure 3.2 Travel Plan Workflow

3.2.2 Security Issues

For every individual part of a MPMSD, the standard security issues are proof of origin, content integrity, confidentiality, non-repudiation of signing etc. These are not the issues of discussion in this chapter. It is assumed that WSS standard addresses these issues for every message in a SOAP envelope efficiently and correctly as claimed. Apart from the security issues of each individual part of a MPMSD, the following special security issues related to a MPMSD as a whole are to be addressed additionally. These issues are the main points of discussion in this chapter.

1) *Part Integrity of a MPMSD* : Apart from the content integrity of each part, we also need the total part integrity of the whole document. The content integrity of all parts individually does not necessarily imply the integrity of the whole document. All parts must remain in order in which they were added to the document. Removal of some parts and reordering of parts should not be allowed.

2) *Reuse of Parts* : Reuse of parts should not be allowed. Suppose, the ordered list of reviewers of a MPMSD is (A, B, C, D) as in the example given in figure 3.2. If D does not like what C has written on the document m_C then D may co-operate with B to have B mark the document directly to D, bypassing C. B can do this by using the document $m_A||m_B$ passed to it by A and reusing it.

3.3 Orchestration and Choreography

Often services are offered by different companies. A Web service composition consists of multiple invocations of other Web services in a specific order. A composition takes the form either of an orchestration or of choreography.

Orchestration describes how Web services interact with each other at the message level, including the business logic and he execution order of the interactions from the view point of the partner controlling of the workflow execution. Orchestration deals with implementation management (i.e. process execution). Orchestration is a private process, controlled by one

party, and defines steps of an executable workflow. Orchestration is about what happens behind interfaces (i.e. process execution) [69].

Choreography describes the interaction between business partners in terms of the sequence of messages that are exchanged in a "peer-to-peer" fashion. In choreography, there is no central control of workflow execution. Choreography is more about what happens between interfaces. It can involve static and dynamically negotiated protocols. In this sense, choreography is a public, abstract process where conversations are composed by equals who define sequences of observable messages [69].

3.4 DPW using SOAP Messages

Document Production Workflow (DPW) is a group work. Therefore, a distributed protocol without any central authority, like the SMTP, POP3 of standard e-mail service, may prop up as a natural choice. In such a distributed approach, the digital document flows from mailbox to mailbox and the reviewers add their comments to the document. This mimics the flow of paper documents in manual offices. We know, protocol binding of SOAP framework allows mail protocols like SMTP, POP3 etc. Without any loss of generality, let us consider that the DPW is implemented using SOAP and WSS with protocol binding to either or both the standard mail protocols for message transmission. A sends a secure SOAP envelope containing a message on original travel plan m_A digitally signed by A, using XML signature to B and so on to C. If A and C collude, C may drop the comment of B and forward the case to D with her strong comment m_C. In a rigid workflow, like in e-Governance, where the reviewers are fixed a priori, such case may be detected by inspecting the absence of B's comment or approval. But in an ad hoc or flexible workflow, like those in many industries, where additional reviewer may be inserted from case to case basis, such collusion may go undetected. For example in our scenario, if manager B thinks that in case of A, comment of another project leader P of a project, where he works in addition to normal duty, is required before approval and forwards the case to P. The comment of adhoc reviewer P can easily be deleted by the colluding parties. Even if physically not deleted, the next reviewer can be bypassed simply by manipulating role and mustunderstand attributes in the security header of the SOAP message. In certain cases reordering of the comments may change the semantic of the case, which may effect the subsequent decisions. Regarding the reuse of parts, as

discussed above, any colluding party can reuse the partial MPMSD available with him or her with another subsequent reviewer in the workflow. This inhibits in itself a new type of attack on SOAP envelope specially in a workflow implementations.

Every reviewer signs not only own comment but also all previous comments up to the original message in the envelope. This can be done by the following steps –

1) Compose the new message.
2) Extract all previous messages from the SOAP envelope.
3) Arrange them in the order of timestamps.
4) Append the new message to the ordered list.
5) Input the composite message to canonicalization method.
6) Generate the message digest.
7) Sign the message and put the signature and other info under *signedinfo* element in security header block.

Since XML signature allows non-contiguous blocks of an XML message to be signed, this may solve the part integrity issue, but the question is how to enforce the user to do so. Moreover, it does not solve the reuse issue. The boundary case that is, if the ultimate receiver D may be a part of collusion. WSS specification claims SOAP envelope security not only during transmission but also during storage to intermediate processing nodes en-route to the ultimate receiver [30]. But it has seen that it is not possible to address these issues in SOAP envelopes directly in the message level. The end-to-end security, envisaged by WSS specification, of the messages contained in a SOAP envelop from the originator of the message A to the final receiver D keeping the intermediaries signed comments intact fails. These issues are more challenging when the envelope routes across many business entities.

3.5 Solution

The main objective of this chapter is to bring into focus some new limitations of WSS on SOAP, specially in workflow environment. However, a solution with a special protocol

exists. Details of the protocol are discussed in [37], [38]. This protocol addresses the security issues of MPMSD mentioned above. The protocol is based on a central arbitration mechanism. It is basically a client/server computing paradigm, where the arbiter is the server. All the cases flowing from one client (reviewer) to another are routed through the arbiter. In case of dispute, the decision of the arbiter, based on the stored evidences, is final and binding. It is discussed in detail in [37] that to address these issues an in-line Trusted Third Party (TTP), called an arbiter, is mandatory. The arbiter serves as the trusted intermediate agent in between the current reviewer and the next. To make a protocol as general as possible, researchers attempt to avoid the use of an arbiter. It is established that to provide non-repudiation with time information, an in-line TTP is anyway necessary [39]. To provide non-repudiation of a digital signature, time of the signature is essential as the key used in the signature may become public at a later time. In this environment, documents are persistent and so non- repudiation of a digital signature is essential. So an in-line TTP will be required. Since an in-line TTP is required for non-repudiation on time of sending or receiving documents, we can en-thrust the TTP, the additional responsibility of an arbiter for production of MPMSD as well. Further, to prevent the reuse of parts, an in-line TTP will evidently be required as immediate detection of such reuse will be necessary to prevent the taking of wrong decisions. When B and D collude to bypass C, C will not be aware of this till much later if no in-line TTP is present [37]. In this protocol, a reviewer submits a comment (part) for a MPMSD to the arbiter and the arbiter adds the part to the MPMSD and present the MPMSD to the next reviewer. The next reviewer can only read the presented multi-part document but cannot modify or add comments to it directly. He can only submit the comment to the arbiter and arbiter adds the comment to the document with proper timestamps and forwards it to the next reviewer.

In a Web service implementation, the BPEL process co-ordinates other Web services. Therefore, it may also act as the arbiter. A reviewer submits the signed comment, which is a SOAP message in an envelope, and the BPEL process extracts the signed message from the envelope and appends it to the envelope containing previous signed messages (MPMSD) in order and then presents modified envelope to the next reviewer. Next reviewer can only read the signed messages, verify the signatures if required but cannot resubmit the presented envelope again. A new envelope containing only her new comment can only be submitted. Therefore, the protocol presented in [37] can be implemented in a BPEL orchestrated workflow which is discussed in the following.

3.6 Discussion

The originator A_1 submits her signed message d_1 to the BPEL process N. Intermediate reviewer A_i receives a SOAP message D_w^{i-1} from N. D_w^{i-1} contains all the previous signed messages $d_1, d_2, d_3 \ldots d_{i-1}$, signed by previous reviewers $A_1, A_2, A_3 \ldots A_{i-1}$. A_i reviews the SOAP message D_w^{i-1}, submits her signed comment $d_i = (m_i, \sigma_i, A_i)$ to N and marks A_{i+1} as the next reviewer. BPEL process N adds the comment m_i to the body section of D_w^{i-1} and σ_i, A_i and other necessary security information in a proper *SignedInfo* element in the security header. N also marks A_{i+1} as the next reviewer by defining the *role* and *mustunderstand* attributes accordingly resulting D_w^i. N then delivers the document D_w^i to A_{i+1}, who in turn submits her comment $d_{i+1} = (m_{i+1}, \sigma_{i+1}, A_{i+1})$ to N. Thus the protocol starts from the originator till the message is delivered to target receiver after being reviewed by intermediate reviewers. The final recipient may be the originator.

3.7 Assumptions

It is assumed that review processes at the reviewers' ends are implemented as Web services and the orchestration is done by the BPEL process. The communication between the BPEL process and the constituent Web services are done by using encrypted SOAP messages, encrypted with the public keys of the recipients or by any other shared keys shared between the BPEL process and each of the reviewers. Since this is not the actual discussion of this chapter, it can be assumed that it as a secured channel. Moreover, it is also assumed that for non-repudiation issues there are necessary tokens for non-repudiation of sending as well as of delivery in place. These were discussed in detail in the original protocol in [37]. Hence the corresponding steps in the protocol are given below where n_i and t_i signify nonces and timestamps respectively.

Protocol Steps:

1) $N \rightarrow A_i : \{N, d_{i-1}, n_i, t_i\}$

2) $A_i \rightarrow N : \{A_i, m_i, \sigma_i, A_{i+1}, n_i, t_i\}$

3) $N \rightarrow A_{i+1} : \{N, d_i, n_{i+1}, t_{i+1}\}$

4) $A_{i+1} \rightarrow N : \{A_{i+1}, m_i, \sigma_i, A_{i+2}, n_{i+1}, t_{i+1}\}$

3.8 Chapter Summary

This chapter discusses certain new limitations of the present version of Web service security standard in a workflow environment for handling document flow in a typical office. SOAP processing model does not discuss these issues. The main focus of this chapter is to show the limitations of WSS and also to show how these issues can be solved by a protocol. From the above discussion it is clear that it is not possible to address these issues within the scope of message level. A central arbitration mechanism is mandatory. In case of orchestrated Web service based workflow, BPEL process can take the responsibility of this arbitration along with coordination of constituent Web services. However, in case of a complex workflow across many business entities, where multiple BPEL processes are to be coordinated, different protocol may have to be used in addition.

The case of choreographed workflow implementations is beyond the scope of discussion of this chapter today but the study will be more interesting. The future research is in this direction. The implementation of the protocol in both orchestrated and choreographed workflows in Web service world is the direction of future research work.

Chapter 4

A Solution to Signature Replacement Attack

2-tuple Digital Signature scheme has two elements: a message and a signature. A tempered message can be verified by the decryption of the message digest, encrypted by the secret key of the signer, with the help of its corresponding public key. On the contrary, if the signature element is replaced then it cannot be verified. This is termed as signature replacement attack. In the case of signature replacement attack, proof of origin is compromised. This chapter mainly focuses on the signature replacement attack. In this chapter a solution for digital signature, resilient to signature replacement attack, is also proposed, where a trusted central arbiter is used as an in-line TTP. However, the central arbiter becomes the main bottleneck of performance. The problem is equally true for XML signature scheme used in Web service security today. This chapter also proposes a solution with a BPEL process which acts as a central arbiter in the proposed special protocol.

4.1 Introduction

Signature is a security mechanism which has been in use for centuries in the realm of paper documents. A signature on a document is the attribute of identification of the signer, approval of the signer to the content and integrity of the content. Signature identifies the signatory with the signed document. It expresses the signer's approval to the contents of the document. Signature establishes signer authentication as well as document authentication where the contents cannot be falsified without detection [42]. The message and the signature in a paper-based signature are tightly bound and hence not separable. Therefore, paper-based signature

can be technically termed as a 1-tuple signature. The security issues addressed by signature are mainly identification, proof of origin, content integrity and non-repudiation. The basic axiom of the signature mechanism is that a particular signature can be produced only by the corresponding signer and cannot be produced by anybody else. Therefore, a signature is uniquely associated to the corresponding signatory. Moreover, the signature is verifiable and can be verified by a third party. Given a signed document, we can identify the signatory by verifying the produced signature with reference to a certified specimen of the signature as done in many manual banking transactions. A signed document itself is the proof that it is originated from the claimed signatory. Any tempering of the content of the message can also be detected by verification. Moreover, at a later point of time, if the signatory repudiates signing the document, it can be established by verifying the authenticity of the signature. Handwritten signature is not a completely error-free mechanism for identification and content integrity of paper-based documents. It can be forged and the content may be tempered in micro-level, which may go undetected in some cases. There are many such forgery cases handled by law enforcing societal agencies. However, with all the weaknesses inhibit in handwritten signature, it is still an acceptable security mechanism in the society.

Digital signatures are generally generated by using public-key crypto system. In this system, a user has a conjugate pair of keys. One key, which is publicly known, is termed as the public key and the other key, which is known only to the user and to nobody else, is termed as secret key of the user. If a message is encrypted with the secret key of the user it can only be decrypted with its conjugate public key or vice-versa. A message encrypted with the secret key of a user can be considered as a digitally signed message, signed by the user, because, this encryption can only be produced by the user in question, as the secret key is known to the user only. The signature can be verified by decrypting the signed message by corresponding public key of the user. Therefore, encryption by the secret key of a user can be considered as the digital signature of the user.

Let, m is a digital message and A is a user.

SK(A) : secret key of A

PK(A) : public key of A

$\{\}_K$: encryption / decryption by key K

Digital Signature : $\{m\}_{SK(A)}$

Verification of signature: $m = \{\{m\}_{SK(A)}\}_{PK(A)}$

Here, $\{m\}_{SK(A)}$ *is the digital signature of A on the message m. Such digital signature can* be termed as 1-tuple signature. Here, signature and the message are tightly bound and not separable and thus the strong property of paper-based signature is properly maintained. But the problem is, encryption and decryption operations in public-key crypto system are time consuming, particularly on large messages. To address this issue a 2-tuple digital signature scheme is proposed as <signature, message> where instead of encrypting the whole message a fixed length message digest is encrypted in the signature element. The message digest is generated by one way hash function.

Let,

δ_m : digest of a message m

h( ) : One way hash function

Signature generation	Signature verification
1. m	1. Receive $< \{\delta_m\}_{SK(A)}, m>$
2. $\delta_m = h(m)$	2. $\delta_m = \{\{\delta_m\}_{SK(A)}\}_{PK(A)}$, signature verified
3. $\{\delta_m\}_{SK(A)}$	3. $\delta'_m = h(m)$
4. $< \{\delta_m\}_{SK(A)}, m>$	4. if $(\delta'_m = \delta_m)$: m is OK : else m is tampered
5. Send $< \{\delta_m\}_{SK(A)}, m>$	

In 2-tuple digital signature, the message and the signature are loosely bound and hence both are separable. On one hand, the 2-tuple scheme provides an efficient digital signature solution and on the other hand it opens up the flush gate of different security attacks. One such attack is XML re-writing attack on SOAP messages [26], [13] and the other one hitherto not discussed in literature, is termed as signature replacement attack. The attack is relevant to any general message not necessarily only on SOAP messages. The attack is possible because of loose binding of signature and message in 2-tuple scheme. This is the major weakness of the scheme.

4.2 SWOT Analysis

SWOT analysis is a strategic planning method used to evaluate the Strengths, Weaknesses, Opportunities and Threat (SWOT) involved in a particular system. It involves specifying the objective of the system and identifying the internal and external factors that are favourable and unfavourable to achieving that objective. A SWOT analysis must first start with defining a desired objective. In SWOT, strength and weaknesses are internal analysis whereas opportunities and threats are external analysis [44]. S-O strategies are to pursue opportunities that are good fit to the system's strengths. W-O strategies are to overcome weaknesses to pursue opportunities. S-T strategies are to identify ways that the system can use its strengths to reduce its vulnerability to external threats. W-T strategies are to establish a defensive plan to prevent the system's weaknesses from making it highly susceptible to external threats. 2-tuple digital signature scheme has its own strengths and weaknesses for digital documents. 2-tuple digital signature is a security mechanism which is designed with an anticipation of the threats likely to be faced by the system.

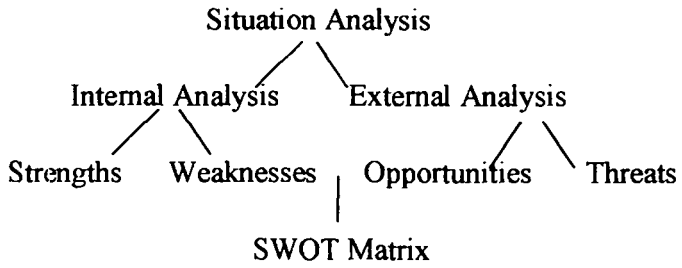


Figure 4.2 SWOT Analysis Framework

	Strengths	Weaknesses
Opportunities	S-O strategies	W-O strategies
Threats	S-T strategies	W-T strategies

Table 4.2 SWOT Matrix

The SWOT of 2-tuple digital signature scheme is discussed as follows -

4.2.1 Strengths

- Main strength of 2-tuple digital signature is that it provides the proof of origin to the message. Signature of the originator of the signed message can be verified by decrypting the signed message digest given in the signature element. Successful decryption of the encrypted message digest, encrypted by the secret key of the signatory, by the corresponding public key of the signatory implies the authenticity of the signature.

- It is temper-proof. If the message is tempered intentionally or unintentionally by a third party during transmission or storage, it can be easily detected. Unintentional tempering or modification may be due to network errors leading to packet loss during transmission. One-way hash function is a publicly known algorithm. A digest of the given message can be regenerated and compared with the digest encrypted in the signature element. Both the digests are identical implies content integrity holds, tempered otherwise.

- It provides user authentication. By verifying a digitally signed token, like the user id, the user can be authenticated. Digital signatures are more secure than using 'user-id' and 'password' to authenticate users. Generally, keys of public-key cryptosystem are comparatively longer and more complex than user designed passwords hence more resilient to eavesdropping.

4.2.2 Weaknesses

- Major weak point of 2-tuple digital signature is that the encryption of a message in public key crypto-system is time consuming. Generally public key encryption and decryption, like RSA, is slower than symmetric key counterparts, like DES.

- Elements of the 2-tuple signature are loosely bound. Therefore, both the message and the signature are separable. As a result, attack may be possible on both the message and the signature separately. If the message is tempered or replaced then it can be

verified by the message digest available in the signature element as discussed earlier. But on the other hand if the signature is tempered or replaced then there is no way to verify it. This weakness is exploited by the signature replacement attack.

4.2.3 Opportunities

- Because of the strengths and the novelty of 2-tuple digital signature scheme based on public-key cryptographic primitives, mainly RSA algorithm, new opportunities are evolving in different domains. One major application is in intra- as well as inter-organizational workflow applications. The use of digital signature in banking workflow applications results in enormous savings in operating costs, improving operating efficiency and quality of customer service. For example, if the banks in India, introduces the digital signature scheme then the fund transfer, clearing of cheques, submission of applications for loans etc. will become quicker and almost instantaneous.

- In relational database applications, digital signatures are typically used to ensured data integrity, non-repudiation and proof of origin. The Oracle security server uses the RSA cryptographic algorithm and message digest 5 (MD5) one way hash function in generating and verifying digital signatures [45].

- In JDK, the most common use of digital signatures is to create signed classes. Operations on digital signatures are abstracted by the signature class. The Java class *Java.Security.Signature* has methods to create and verify a signature. The J2EE technology supports Web Service Security which can also be used to create digital signature [46].

- In XML security, XML signature provides a useful means of expressing a digital signature on XML data. The main standards of XML security are XML canonicalization, both inclusive and exclusive, XML encryption and XML signature. XML signature provides all the services of 2-tuple digital signature system for XML data. It's defined on the top of standard cryptographic functions, like RSA, MD5 etc., but it does not provide these functions on its own.

- XML Signature finds multiple uses for Web Services security. Since Web Services communicate to each other through SOAP messages, the XML digital signature can be used in message integrity and non-repudiation [43]. When SOAP routing occurs, XML digital signature can be used for workflow.

4.2.4 Threats

- As mentioned earlier, the main weakness of 2-tuple digital signature scheme is that the signature and the message are loosely bound. Only logical connection between the message and the signature is the digest of the message in the signature element irrespective of the position of the message signed in the document. This is posed as a threat and exploited by the well-known XML rewriting attack on SOAP messages [13], where original message is pushed to a different position and the replacing new message will co-exist with the original message in the same SOAP message. SOAP is an XML message. Therefore, it is basically a tree. In a SOAP message where signature element is a child of a security header element, which in turn is a child element of SOAP header element. The message signed appears as a descendent of SOAP body element. SOAP header and body elements are children of the root envelope element. Logical link between the message and the signature is the unique ID of the message incorporated in the signature element. Now an attacker can replace the message by another message and wrapping the original message under a bogus element. This will fool the verification process as the logical link of message ID still holds. A detail of this attack is available in [13], where a formal solution of XML rewriting attack is given. It incorporates the context of the signed message, captured as productions of regular tree grammar, in signature.

- In a certificate-less signature scheme, Key replacement attack is another type of threat where a third party can replace the user's public key or secret key without knowing the user's private key which has been issued by the Key Generation Centre (KGC) [40].

- The signature replacement attack that is going to discuss in detail in the next section, is another attack exploiting the same weakness of 2-tuple digital signature system. In

this attack signature element is replaced by the attacker and by doing so claim the ownership of the message.

4.3 Signature Replacement Attack

The main objective of the 2-tuple digital signature is to provide security to the message and the signature separately. The signature replacement attack is the replacement of a digital signature in a digital document by an attacker without altering the content of the message. This attack is possible only because the main weakness of 2-tuple digital signature, as discussed earlier, is that it is loosely bound. Based on this weakness, the signature replacement attack has identified. When the message element of the 2-tuple digital signature is replaced by another message, it can be easily verified by content integrity checking. Existing signature scheme is strong enough to address this issue. On the contrary, if the signature element is replaced by another signature, claiming the ownership of the content of the message, present signature scheme has no mechanism to detect it during verification. In this case, proof of origin will be consequently compromised. This attack is not so trivial and is hitherto not discussed in literature.

Suppose, in an office system, an employee A submits an application to B where A is the originator and B is the receiver. The originator A sends the digital document to the receiver B with his digital signature. After receiving the digital document from A, B verifies the message and simply replaces the message. In this case message is obviously tempered, which can be verified by content integrity and proof of origin checking. But, on the other hand if B simply replaces the A's digital signature by his signature keeping the content of the message remain same and claiming the ownership of the message, there is no way to verify the forgery. This is possible because, message digest can be generated by any third party and the one-way hash function used to do so is public.

Main observation is that in 2-tuple signature scheme, the relation between the signature and the message elements is not symmetric. This means that by the mechanism of incorporation (encryption by the secret key) of the message digest in the signature, relation of the message to the signature is set and is not forgeable and also is verifiable by a third party. But on the other hand, there is no mechanism by which a relation can be set from signature to the message, which is not forgeable and is verifiable. Since 2-tuple digital signature is loosely

bound and hence message and signature are separable, separate attacks are possible on both the message and the signature.

4.4 An Example Scenario

Signature replacement attack discussed in this chapter is not so trivial. In certain domains it may be posed as a major security issue of concern. Let us take a general scenario from a patent office. A patent protects the methods and processes that make things work. A patent for an invention is the grant of a property right to the inventor issued by the patent office. A Patent is required for giving protection to inventor's new idea or invention. Assume that, while sending the patent application to the patent office, the inventor must digitally sign the application and then send it to the destination. The digital signature in the patent application form is required because after sending, the inventor will claim the ownership of the application.

The patent office shall be in a position to effectively handle the filing of patent applications and other related documents received in the office. An inventor sends a patent application, which is digitally signed, to the patent office for publishing his new idea or invention. In that patent office, a receiver will receive that patent application for verification. Now, the receiver of the patent office has the opportunity of claiming the invention by replacing the digital signature in the patent application submitted. In such case the tempered signature replaced by a receiver of patent office will be treated as a valid signature. So, a signature is tempered keeping the content of the patent application remaining same and the receiver of the patent office or his collusion partner will become the inventor. In this scenario, a signature replacement issue is important which cannot be solved by the available 2-tuple digital signature scheme. If the knowledge is considered at best, this attack is not discussed in literature, so also the solutions. This chapter proposes a solution to resolve this issue.

4.5 XML Signature and SOAP

XML signature is the specific syntax used to represent a digital signature over XML data. XML signatures are digital signatures which are generally used in XML transactions. As XML becomes a vital component of the emerging electronic business infrastructure, we need

trustable, secure XML messages to form the basis of business transactions. One key to enabling secure transactions is the concept of a digital signature, ensuring the integrity and authenticity of origin for business documents. XML signature is an evolving standard for digital signatures in XML world [13], [22].

On the other hand, Simple Object Access Protocol (SOAP) is used for communication among different Web Services. SOAP [12] messages flow from originator to an ultimate receiver through a SOAP message path. SOAP messages are nothing but XML documents. SOAP is the combination of XML and HTTP which is used for message exchanges. SOAP features document-based communication among Web services. Document-based communication allows the integration of loosely coupled services. A simple SOAP message has mainly two parts: *Header* and *Body*. *Header* includes service invocation, invocation results, sender's credentials, and response type and so on. *Body* includes XML representation of service invocation request or response. SOAP implementations automatically generate the SOAP Header and provide mappings between the contents of SOAP message bodies and data structures in the host language. Therefore, a simple SOAP message can be represented using a tree like structure which has shown in figure 4.5 [26]. Details of XML Signature and SOAP are available in chapter 2.

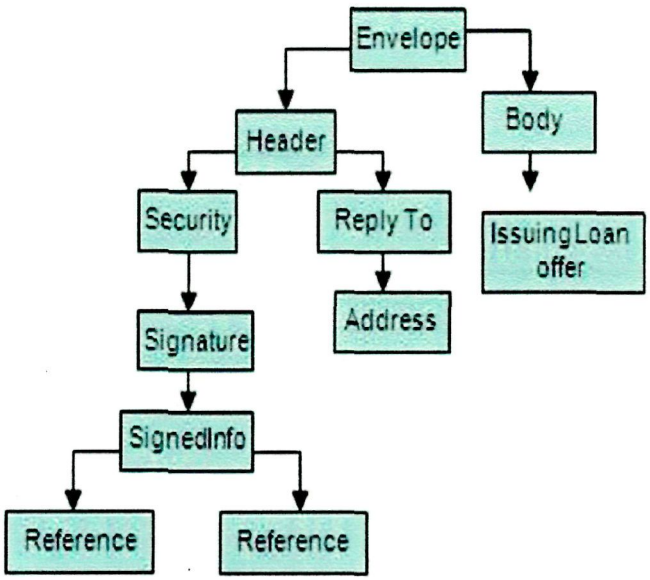


Figure 4.5 Tree Representation of a SOAP Message

4.5.1 XML Signature on SOAP Messages

XML signature can be used on SOAP messages. Use of XML signature on SOAP messages made it more popular. This popularity attracts more attackers to design new attacks on SOAP messages. Parts of a SOAP message can be encrypted and signed using XML digital signature. The signature is used by the receiver to check integrity of the message and authenticity of the sender. SOAP messages are carrying information from one Web service to another. In this context security plays an important rule; therefore a Web service security specification emerged [26]. To ensure authenticity and non-repudiation a signature mechanism is provided based on the XML digital signature specification.

4.5.2 SOAP Security

XML signature is an implementation of 2-tuple digital signature which incorporates SOAP security. Since 2-tuple digital signature has its strength and weaknesses, so the same will show in case of SOAP security also. Since 2-tuple digital signature is loosely bound and the message and the signature can be attacked separately, so likewise possibility of attack will be there on SOAP messages. It uses XML signature for signing non-contiguous parts of a SOAP message. Therefore, all the limitations of digital signature are also applicable to XML signature in SOAP security. It allows multiple security headers with the same name to exist in the same SOAP message. This creates a pitfall and can be exploited by an attacker. It does not propose any new security technology. However, it specifies how the existing security technology can be used to secure a SOAP message exchange. It encompasses many other standards like XML Digital Signature, XML Encryption, X.509 certificate, Kerberos ticket etc. For this reason, the specification is quite complex [26].

4.5.3 Attacks on Signed SOAP Messages

XML signature allows non-contiguous object of an XML dataset to be signed separately. The signed object is referenced by an indirection (URI) from the <Reference> element of the signature. This indirect referencing gives no information about the actual location of the

signed element. This leads to rewriting attacks, where an object can be relocated and the signature still remains valid [16].

1) *XML Rewriting Attack:* The XML rewriting attack is an exploitation of the XML tree by reordering the existing nodes or by wrapping nodes without altering the signature. This attack is depicted in the figure 4.5.3 (a) and 4.5.3 (b). On the left side of the figure 4.5.3 (a), is a valid message with a signed body. On the right side of the figure 4.5.3 (b), the attacker wraps the existing body into a bogus header, making this body meaningless for the receiver and creates its own element namely "newBody". According to SOAP specification, the "newBody" element is not required to have an identifier "Id". This signature is still valid because the referenced element "#1" is still present and not changed [13].

2) *XML Signature Replacement Attack*: XML signature on a SOAP message is an implementation of 2-tuple digital signature scheme. The signed message generally resides as a sub-tree of Body element whereas the signature along with its metadata captured in different elements as given in figure 4.5 resides as a sub-tree of a security header element, under a SOAP envelope. Therefore, such loosely bound signature can be easily replaced as discussed earlier.

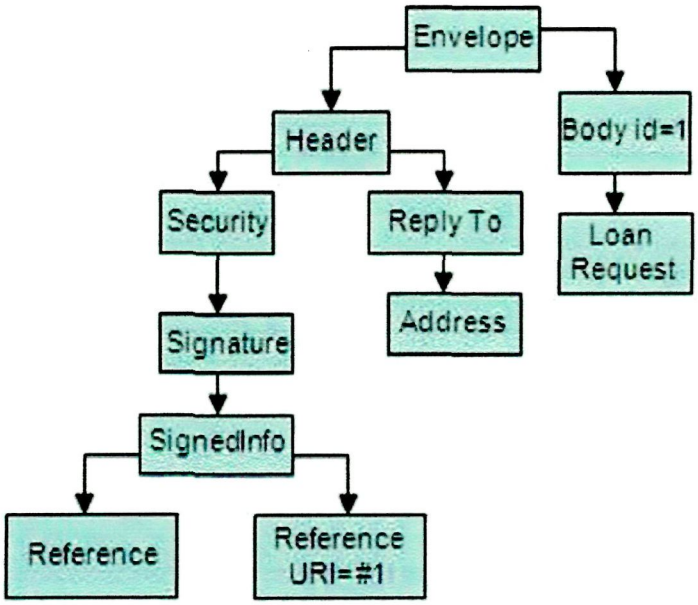


Figure 4.5.3 (a) SOAP Message with a Signed Body

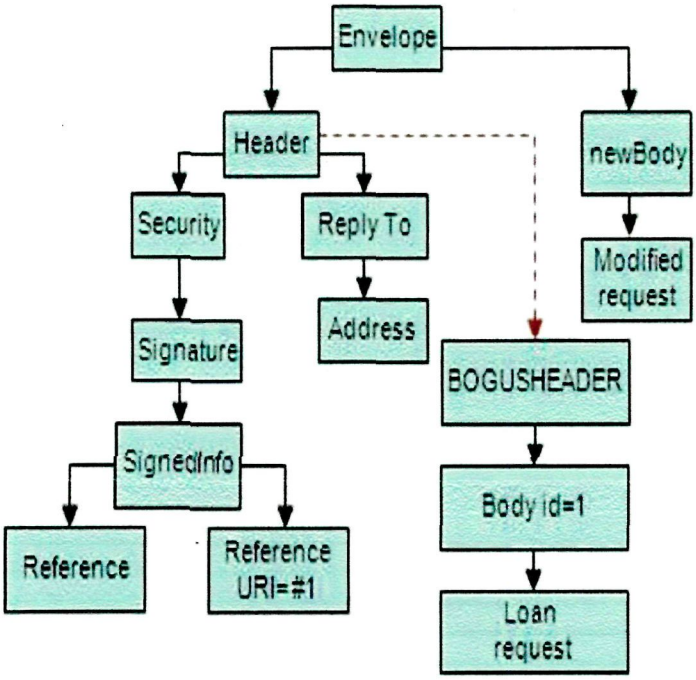


Figure 4.5.3 (b) Rewriting Attack with a newBody

4.6 Solution

The main objective of this chapter is to bring into focus the weaknesses of 2-tuple digital signature scheme which paves the way for signature replacement attack, which is not discussed in the literature properly. Giving a solution to this problem is not the main goal in this chapter. However, it seems that no solution is possible keeping the loose binding of the signature and the message. However, immediately what comes to mind as a counter-measure is to create a digital envelope encapsulating the signature and the message.

Digital envelope is simply an encryption of data with the public key of the recipient which reaches to the recipient. Digital envelope is designed to address the confidentiality issue. As the message and the signature are encrypted with the public key of the recipient, it cannot be decrypted by the third party in the middle. Therefore, digital envelope works as a counter-measure of replacement attack during transmission but not during storage. Suppose, a digitally signed message is sent to B in a digital envelope. After receiving the envelope, B will verify the message and the signature. After verifying, B himself can temper the signature and claim the ownership of the message. So, in this case signature replacement attack is done during storage. From this discussion it is clear that, during storage digital envelope cannot

protect the signed message from signature replacement attack. It means that, digital envelope fails to give the solution on signature replacement attack.

However, a solution with a special protocol exists. This protocol is based on a central arbitration mechanism. It is basically a client/server computing paradigm, where the arbiter is the server. All the messages flowing from the originator to the next receiver are routed through the arbiter. To address these issues an in-line Trusted Third Party (TTP), called an arbiter is mandatory. The arbiter serves as the trusted intermediate agent in between the originator and the receiver. In the workflow environment documents are persistent and so non-repudiation of a digital signature is essential. In this protocol, an originator submits an application to the arbiter and the arbiter presents the application to the next reviewer or the receiver. The receiver can only read the application and verify the signature if it is needed but cannot modify it directly. He can only submit the comment to the arbiter and arbiter adds the comment to the document with proper timestamps and forwards it to the next reviewer if necessary [41]. Therefore, an in-line TTP may be used in the protocol to meet the objectives of signature replacement attack which gives the bottleneck of performance.

4.7 Discussion

In a Web Service implementation, the BPEL process co-ordinates other Web services [29]. In this case, the BPEL process acts as an arbiter and other co-ordinated Web Services act as clients. An originator submits the signed document which is a SOAP message in an envelope, and the BPEL process extracts the signed message from the envelope and appends it to the envelope containing previous signed messages in order to present modified envelope to the next reviewer or the ultimate receiver. Next reviewer or the ultimate receiver only can go through the signed messages, verify the signatures if required but cannot re-submit the presented envelope again. A new envelope containing only his new comment can only be submitted. Therefore, this protocol presented in [37] can be implemented in a BPEL process which will meet the objectives of security issues particularly in signature replacement attack [41].

4.8 Chapter Summary

This attack of signature replacement gives us another line of thinking. In paper based signature the message content and the signature are tightly bound. Hence, the questions of signature replacement attack is not that much relevant. In public-key based 2-tuple digital signature, the signature and the message content are loosely bound. In one hand this signature scheme is adventitious from efficiency point of view but on the other hand it invites vulnerability from security point of view. Present digital signature standard is not resilient to signature replacement attack. Therefore, a solution has to be searched in protocol level as the one proposed in this chapter with a central arbiter. The central arbitration mechanism is mandatory to address such issues. This chapter highlights a solution with a central arbiter as an in-line Trusted Third Party (TTP) mechanism.

This weakness of digital signature might have skipped the attention of the researcher thinking it as a trivial issue. After thorough investigation and SWOT analysis it is found out that in certain situation this weakness may be posed as a threat which the attackers may capitalize on. Since the message level 2-tuple digital signature scheme is incomplete, the business solutions today adopt socio-technical steps, like notifications, in protocols as *counter-measures*, particularly in workflows. The future work includes investigation of rationale behind socio-technical steps in protocols involving digital signature. The future works also include detail study on signature replacement attack and its counter-measures in both orchestrated as well as choreographed Web services.

Chapter 5

Synchronization of Authorization Flow with a Work Object Flow

This chapter presents the issue of synchronization of authorization flow with work object flow in a document production workflow environment. The chapter has shown how a work object flow is synchronized with the authorization flow using a central arbiter in Web service paradigms. The co-ordination of Web services is done using WS-BPEL which supports orchestration and XACML provides authorization for Web services. The synchronization is achieved by exploiting the obligation provisions in XACML.

5.1 Introduction

Synchronization of authorization flow with the workflow is a fundamental security requirement in a workflow environment. In case of automatic time-bound execution of tasks, issue of authorization flow is addressed by temporal constraints in the definition of tasks in the workflow. It is suitable for time-bound production workflows. In a general e-Business scenario, authorization flow is based on events not on temporal constraints. Authorizations for certain privileges are granted to a subject on some objects based on the occurrence of an event and revoked automatically on occurrence of the other conjugate event. The most common conjugate pair of events in an e-Business scenario is *send* and *receive*. As soon as the work object is sent to the next task, the current role and hence the task loses full or partial

privileges to the work object, whereas next role gains certain privileges on the work object to act upon by the task assigned to the role.

Let us take the case of an office. After receiving a request document, an office reacts to it, reactions are recorded as comments and finally a response document is sent to the requester. In a typical office, a worker, called the originator, creates a document and sends it to the another worker, who examines the case with respect to existing rules, regulations and precedents, adds his own comment along with the context and forwards it to the another worker and so on. As a result, we get a composite document, composed of an ordered list of comments contributed by the originator and the reviewers during the process. This workflow is termed as Document Production Workflow (DPW) [41]. The output of this workflow is the ordered list of comments, called Multi-Part Multi-Signature Document (MPMSD) [37]. In a MPMSD, the first part is the request document, the last part is the response document and the other parts in between are the reactions of the reviewers. A reviewer can read the previous parts but cannot modify the content of any of the previous parts, even cannot reorder or drop any of the previous parts [37]. The work object of such a workflow is a multi-part object. Each part is the output of a task. A role assigned to the task is granted authorization to execute the task only when the previous parts of the work object are in place. Previous user looses the write privileges as soon as the work object is sent to the next user. The present user gains the privileges to read previous parts and add a new part to the work object as soon as she receives the work object.

A comprehensive study to the MPMSD protocol with a central arbiter was done in [37]. The main aim of this chapter is to study the protocol in Web service domain, specially the issue of synchronization of authorization flow with the work object flow, both in architecture level and protocol level, in a Document Production Workflow (DPW) environment.

5.2 A Scenario

This is a Document Production Workflow (DPW) scenario common to many offices. An office worker, A submits an application; m_A regarding his leave sanction for approval to the Head, B of the department, where he is working. B verifies the leave rules and gives his comment, m_B and forwards it to the Registrar, C. C verifies all the leave status and gives his

comment, m_C on the application and forwards it to the Director, D. D approves the leave application with the addition of his approval note, m_D. The multi-part document finally may go back to the originator A and the original multipart document is stored in a folder. The flow of the document is recorded in log-books. This is a case of the travel plan workflow. The DPW of the scenario is available in [41].

In a manual system, it is the same paper document that is passed around and the proof that it has come through the proper channel with the addition of series of comments followed by the signatures of the reviewers. As soon as an employee submits a travel plan request, the request document forms the first part of the work object. After sending the document to the next reviewer, the employee looses read, write privileges on the work object. The Head is granted to execute task verify leave status of the employee, add a part on the document and sends it to the next reviewer and so on.

5.3 Security Issues

The Security issues of DPW are like part integrity of a MPMSD, reuse of parts and authorization flow. Details of the issues like part integrity and reuse of parts are discussed in [37]. However, the issue of synchronized authorization flow is not clearly discussed. According to authorization flow, access right to a MPMSD moves from one reviewer to the next synchronized with the flow of the work object, i.e. MPMSD. Only the latest reviewer can read the document and can add comments to it. A reviewer can only read the previous parts. She is not allowed to modify the contents of any of the previous parts. Even the latest reviewer of a part should not be allowed to modify her own comment after signing and forwarding the document to the next reviewer. This chapter mainly focuses on this issue particularly in the parlance of Web Services. A protocol for MPMSD production with a central arbiter was proposed in [37], [38]. This protocol addresses the security issues raised in this section. The challenge in the Web Service domain is how to implement the protocol using the current industrial standards, like BPEL and XACML and ensure the synchronization of authorization flow with the MPMSD flow.

5.4 MPMSD Production Protocol

The responsibilities of generation, storage and flow control of MPMSD lies with a central, neutral, trusted and strong arbiter. First, an employee submits a travel plan request to the arbiter. The arbiter will verify the identity of the employee whether she is a valid user or not. As soon as the employee submits an application to the arbiter, she loses modify or delete privileges and gains read privilege depending on arbiter policy. The arbiter will present a multi-part document to a reviewer. The reviewer will submit her comment on the document to the arbiter along with the user-id of the next reviewer. The next reviewer can gain read privileges of previous comments as well as add her comment on the document but loses modify or delete privileges on the document. The arbiter will verify the comment and add it to the document as a part and then present the document to the next reviewer and so on. During the process it will also generate the non-repudiation evidences of submission of a comment and of delivery of a document. It is shown in the figure 5.4 (a), where I_d is the unique identifier of the document given by N [37].

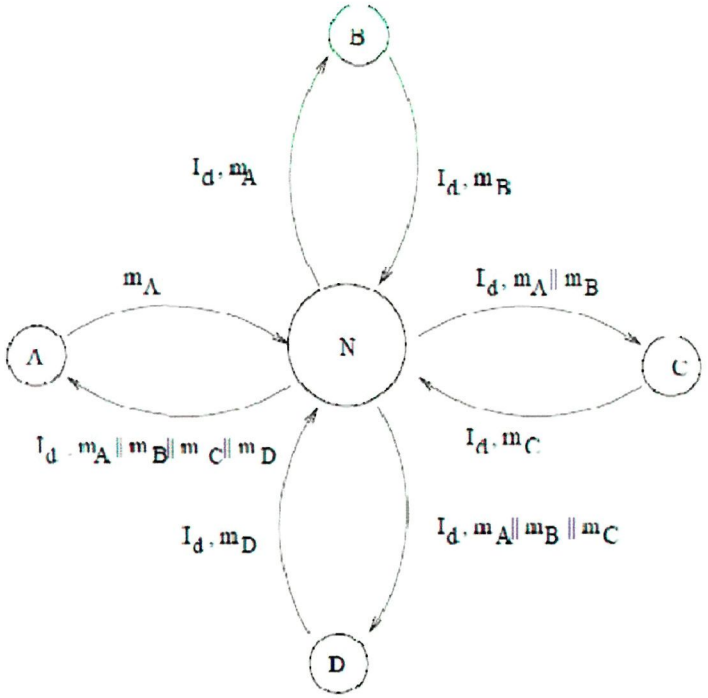


Figure 5.4 (a): Generation of MPMSD

Summary: A_i reviews the document d_{i-1} , submits her comment m_i to N and marks A_{i+1} as the next reviewer. N adds the comment m_i to d_{i-1} giving d_i and then delivers the document d_i to A_{i+1}, who in turn submits her comment m_{i+1} to N which has shown in figure 5.4 (b) [37].

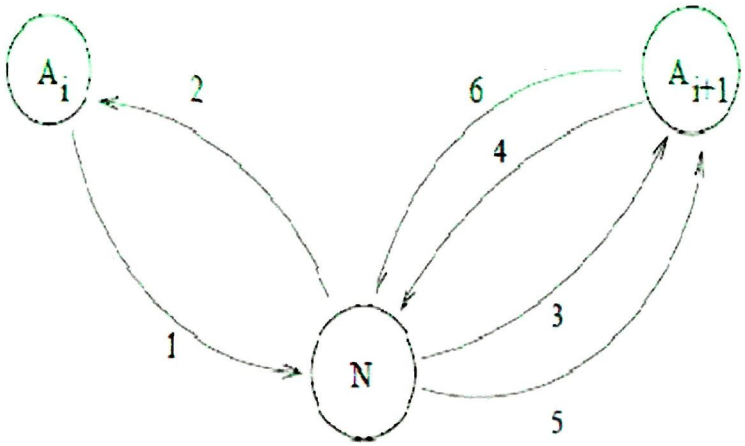


Figure 5.4 (b): MPMSD Protocol

5.4.1 Protocol Steps

The protocol steps for MPMSD production are as follows:

1. $A_i \rightarrow N : \{ A_i, m_i, I_d, A_{i+1}, d_i\}$
2. $N \rightarrow A_i : \{ N, d_i \}$
3. $N \rightarrow A_{i+1} : \{N, d_i, m_i\}$
4. $A_{i+1} \rightarrow N : \{ A_{i+1}, d_i\}$
5. $N \rightarrow A_{i+1} : \{ N, d_i\}$
6. $A_{i+1} \rightarrow N : \{ A_{i+1}, m_{i+1}, I_d, A_{i+2}\}$

Actions at Each Step

1. $A_i \rightarrow N$: The originator A_i submits a document d_i with her comment m_i and marks it to the next reviewer A_{i-1} by using an identity I_d of the next reviewer and then sends it to the arbiter N.

2. $N \rightarrow A_i$: The arbiter N then verifies the originator A_i whether she is a valid user or not for accessing the document d_i.

3. $N \rightarrow A_{i+1}$: The arbiter N then sends the document d_i to the next reviewer A_{i+1} with the comment m_i of A_i.

4. $A_{i+1} \rightarrow N$: The next reviewer A_{i+1} sends a request to the arbiter N for the permission of accessing the document d_i.

5. $N \rightarrow A_{i+1}$: The arbiter N then verifies the identity I_d of the next reviewer A_{i+1} and then gives permission to access the document d_i which is sent by the originator A_i.

6. $A_{i+1} \rightarrow N$: The next reviewer A_{i+1} submits her comment m_{i+1} on the document d_i and marks it to the next reviewer A_{i+2} with an identifier I_d and so on.

5.5 WS-BPEL for DPW

In general, Web service, under Service Oriented Architecture (SOA) framework, is the state of the art technology for implementing workflow. The co-ordination of Web services is done in two ways: either by choreography or by orchestration. WS-BPEL is the standard which supports orchestration. The central arbitration mechanism discussed in section 5.4 is a form of orchestration. Therefore, an ideal technology to implement DPW with central arbitration mechanism is WS-BPEL. Here, WS-BPEL process works as arbiter N and Web services implement different review processes accomplished by different reviewers as mentioned in the protocol. WS-BPEL process co-ordinates all the Web services and also controls the flow of the work object.

Business Process Execution Language (BPEL) is an XML-based language to specify business processes that orchestrate the operations of several Web services. WS-BPEL is

designed to describe business processes in a structured way. The business logic is expressed as a group of activities and performed by invoking Web services [48]. The <process> element is on the top level of BPEL specification. The attributes of <process> element specifies the process name and related namespaces being referred to. The <partnerLinks> element indicates the external Web services to be invoked from this process. The <variable> element defines the data variables involved in this process. The <sequence> element contains sequentially executed set of activities. Each step in a WS-BPEL process is called an activity. The <invoke> element that allows the business process to invoke a one way or both way (request-response) operation offered by a partner. The elements may contain one or more basic elements such as <receive> and <reply> element which define the message flow in a process. A synchronous BPEL process is one which we call and wait for the reply before proceeding further. On the other hand, an asynchronous BPEL process is one which we call and instead of waiting for the reply continue further proceeding. Both the processes first wait for the initial message using a <receive> element. A synchronous BPEL process will return a response after completing the process. Therefore, we use a <reply> element at the end of the process. But, an asynchronous BPEL process does not use the <reply> element. If such a process has to send a reply to the client, it uses the <invoke> element to invoke the callback operation on the client. In WS-BPEL process for DPW, an originator sends a request to the arbiter and waits for the response until and unless arbiter gives reply. After getting reply from arbiter, the originator will proceed further for the next process. The scenario that discussed in section 5.2 is synchronous only. However, in other some scenarios of DPW, it can be asynchronous as well. The detail of WS-BPEL specification is available in [51].

In the travel plan scenario the whole process can be divided into five activities:

- The employee applies for Travel Plan (travelRequest).
- Head verifies the leave rules on employee's travel plan request (travelAReview).
- Registrar verifies the leave status on originator's travel plan (travelBReview).
- Director approves the travel plan (travelApprove).
- The application response is sent back to the originator (travelResponse).

The process orchestrates the operations of four Web services:

- A Web service that provides the operation (travelRequest) to apply the travel plan.
- A Web service that provides the operation (travelAReview) to verify leave rules.
- A Web service that provides the operation (travelBReview) to verify the leave status.
- A Web service that provides the operation (travelApprove) to approve the travel plan decision.

The WS-BPEL implementation for DPW can be worked out as discussed in the following. The <partnerLinks> element indicates the participators or reviewers in the travel plan application process. The <variable> element defines the variable name as 'travelRequest' for travel plan application. In <sequence> element, all the steps involved in travel plan request will be invoked in an ordered sequence. An employee submits a travel plan request to the arbiter. The <receive> element receives a travel plan request from the employee. The <assign> element will manipulate the data variables and the <copy> element will copy data between <from> and <to> element by assigning the variable 'travelRequest' to any other variable. The reviewerA review the travel plan request invoked by the <invoke> element on the employee's document. The <assign> and <copy> element will concatenate the variables 'travelRequest' and 'travelAReview' using 'concat' function. The reviewerB review the travel plan request which is invoked by the <invoke> element. The <assign> and <copy> element will concatenate the variables 'travelAReview' and 'travelBReview' using the 'concat' function. The approver approves the travel plan request. The approver makes a decision for the travel plan request which is further invoked by the <invoke> element. The <assign> and <copy> element will concatenate variables 'travelBReview' and 'travelApprove' using 'concat' function. Finally, the travel plan response is sent back to the employee by the <reply> element.

5.6 XACML for Authorization Flow

The eXtensible Access Control Markup Language (XACML) is an XML based language which is required to make authorization decisions. The decision may be permit, deny, indeterminate or error. The XACML architecture [47] specifies the implementation of a Policy Enforcement Point (PEP), which is an entity that performs access control by enforcing authorization decision. A Policy Access Point (PAP) is an entity that creates policies or policy sets. A Policy Decision Point (PDP) is an entity that evaluates applicable policy or policySets to renders an authorization decision. A Policy Information Point (PIP) is an entity which is having information about the attributes of subject, resource, action and environment. The Context Handler receives the access request from PEP and converts the access request context to the XACML context. The data flow model of XACML architecture is as follows –

At first, the PAP creates a policy. At request time, an access request arrives at the PEP and is send to the Context Handler. The Context Handler determines resources to be accessed, collects all required attributes of subject, resource, action and environment from PIP and forwards them to the PDP. PDP then acquires the policy from the PAP, evaluates the relevant policy and returns the access decision to the PEP through Context Handler which proceeds to enforce the authorization decision. The PDP sends the authorization decision to the PEP through Context Handler with some obligations. The PEP fulfils the obligations. If access is permitted, then the PEP permits access on the resource; otherwise, it denies access [23], [50].

The authorization flow using XACML in DPW play an important role while grant and revoke privilege on a work object. Considering the example of travel plan application, during authorization flow a reviewer is granted read and adds comment while she is revoked write, modify or delete privileges. An employee A_i submits a travel plan request to the arbiter N. The arbiter N will first verify the identity of A_i whether she is a valid user or not. As soon as the travel plan request is submitted to the arbiter, the employee is revoked modify or delete privileges. She may be granted read privileges depending on the arbiter policy. The next reviewers such as reviewerA and reviewerB are granted read previous comments and add their own comments on the travel plan request where they are revoked modify or delete privileges. Similar is the case of approver where she is granted to verify all the previous comments and give his own comment for approval but she is also revoked modify or delete privileges.

5.6.1 XACML Protocol

The protocol steps for XACML architecture are as follows:

1. PAP → PDP : {PAP, ps_i}
2. A_i → PEP : {A_i, a_{req}, r_1}
3. PEP → CH : { PEP, a_{req}, r_1}
4. CH → PDP : { CH, a_{req} , $xacml_{req}$}
5. PDP → CH : {PDP, s_{attr}, r_{attr}, a_{attr} }
6. CH → PIP : { CH, s_{attr}, r_{attr}, a_{attr} }
7. PIP → CH : { PIP, s_{attr}, r_{attr}, a_{attr} }
8. CH → PDP : { CH, s_{attr}, r_{attr}, a_{attr}}
9. PDP → CH : {PDP, r_1, a_{dec}}
10. CH → PEP : {CH, r_1, a_{res}}

Actions at Each Step

1. PAP → PDP: The PAP writes policySets ps_i and makes them available to the PDP.
2. A_i → PEP: An originator or reviewer A_i sends an access request a_{req} to add to the PEP for access an object r_1.
3. PEP → CH: The PEP then sends the access request a_{rcq} to the Context Handler for an authorization decision.
4. CH → PDP: The Context Handler converts the access request a_{rcq} to an XACML request $xacml_{rcq}$ and sends it to the PDP.
5. PDP → CH: The PDP then sends a request to the Context Handler for the information about the attributes of subject, resource and action (s_{attr}, r_{attr}, a_{attr}).

6. CH → PIP: The Context Handler then sends a request to the PIP for information about the attributes of subject, resource and action (s_{attr}, r_{attr}, a_{attr}).

7. PIP → CH: The PIP obtains information about the attributes of subject, resource and action (s_{attr}, r_{attr}, a_{attr}) and then returns them to the Context Handler.

8. CH → PDP: The Context Handler then sends the information about the attributes of subject, resource and action (s_{attr}, r_{attr}, a_{attr}) to the PDP which will be matched with the policySet ps_i available in PDP.

9. PDP → CH: The PDP then returns an authorization decision a_{dec} on the object r_1 to the Context Handler.

10. CH → PEP: The Context Handler finally returns the access response a_{res} on an object r_1 to the PEP. The PEP fulfils the obligations. If access is permitted, then the PEP permits access on the object r_1; otherwise, it denies access.

5.7 Synchronization

In the earlier sections of this chapter, the work object flow using BPEL where the flow of an object is controlling by the WS-BPEL process is discussed. The authorization flow using XACML where an authorization decision is given to an access requester is also discussed. The reviewers can access the work object by applying the granted and revoked privileges. Now, in this section, the solution of synchronization of authorization flow with a work object flow has shown in two levels – the architecture level and the protocol level. In the conceptual architecture level, a 3-tier architecture is proposed. In the protocol level, the merging of the protocol for work object flow and the protocol for authorization flow are discussed.

5.7.1 Architecture

A three tier architecture consists of three layers namely - presentation layer, logic layer and data layer. The presentation layer consists of view which contains GUIs (Graphical User Interfaces), the logic layer consists of processes to manipulate data and data layer stores data. The logic layer is again split into two – client logic and arbiter logic because the data

manipulation will be done by the client and the arbiter co-operatively. The client and the arbiter will communicate over a network using some protocols. The components of the architecture are described below –

The View

This part of the architecture gives basically the GUIs (Graphical User Interfaces) [36] of the system. View comprises of various interfaces through which a reviewer can interacts with the client logic module to review a document and adds own comment and forwards it to the next reviewer. Through this tier a user interacts with the workflow agent of the client logic. The interaction may be different for different roles.

The Client Logic

This module is populated by client side objects with attributes and methods. The objects in this module are termed as agent. The main agent found in this module is –

MPMSD client: This agent is responsible for client activities contain an object which submits an access request to the MPMSD arbiter for authorization decision. The responsibilities of this agent are – request the arbiter to get authorization decision, get the response from the arbiter as an authorization decision, and display the document in the GUIs module.

The Arbiter Logic

The arbiter is the central hub of the architecture. It certifies and authenticates subjects and objects, manages storage and retrieval of objects, manages authorizations of subjects on objects, time-stamping etc. The services of the arbiter are provided by the following components –

MPMSD arbiter: This component includes – forwarding a MPMSD to the next reviewer (client), addition of the part submitted by the client to the MPMSD under production, sending evidence of submission (NRS) of parts to the client, recording document flow in the logbooks, time-stamping the evidences of occurrences of events in the communication (sending, receiving, authoring etc.), authorization flow management during the communication with other components. In our scenario, since MPMSD arbiter is acting as a

TTP (Trusted Third Party), so without any loss of generality, in addition, it can also function as PEP and obligation handler. This is how synchronization of authorization flow with work object flow can be achieved.

Storage Manager: This component stores and retrieves the document in and from the data storage.

The components like CH, PDP, PAP and PIP are as in the standard XACML architecture which is already discussed in section 5.6.

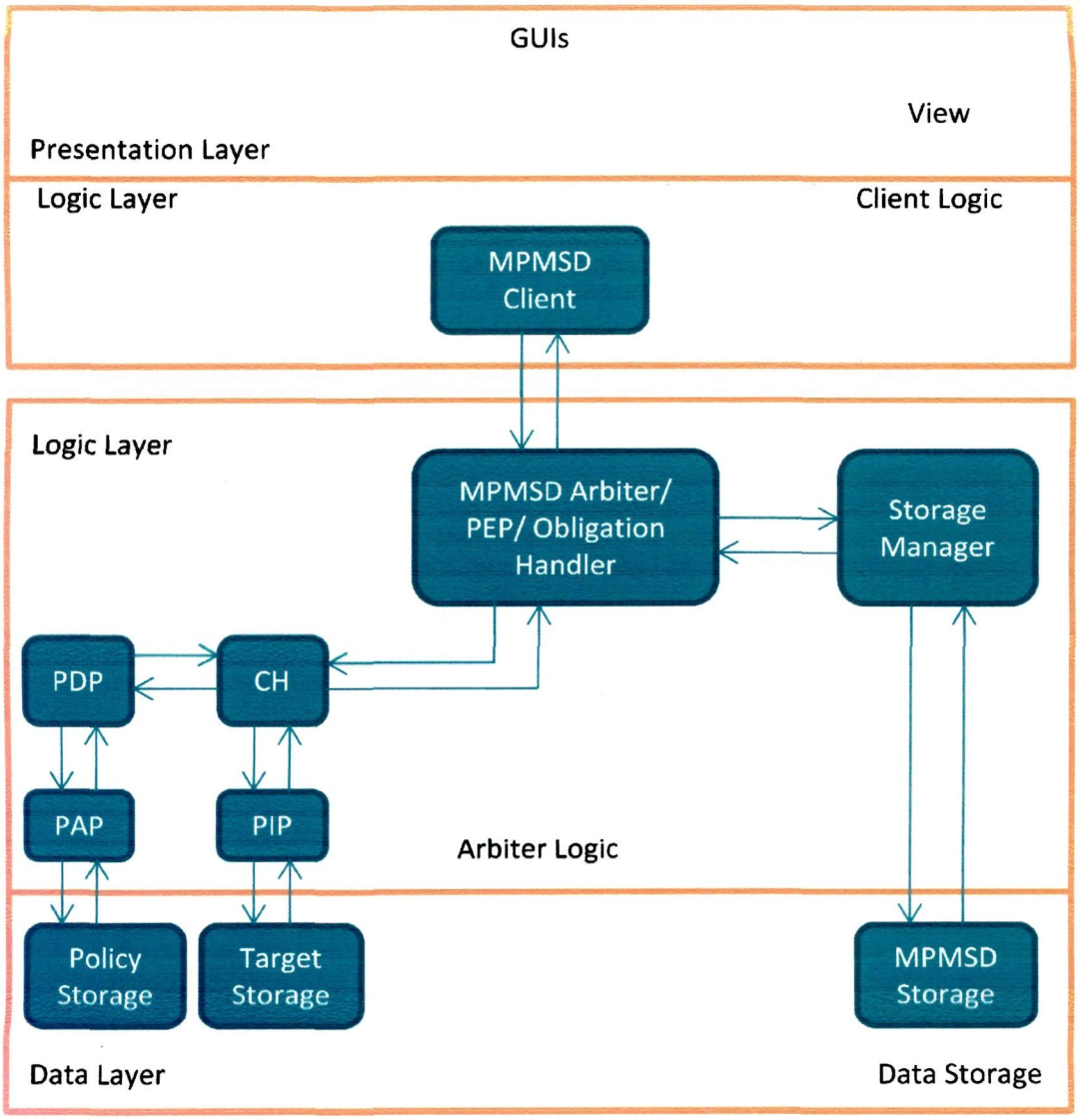


Figure 5.7: The Three Tier Architecture

The Data Storage

This layer is basically a repository of office objects [35]. *MPMSD storage* stores multi-part documents which is managed by storage manager. The *Target storage* stores the attributes of subjects, resources and actions along with their values as per the XACML architecture which is managed by PIP. *Policy storage* stores authorization policies managed by PAP.

5.7.2 Protocol

The synchronized protocol steps of both XACML and DPW are as follows:

1. $A_i \rightarrow N$: {A_i, add, d_i, m_i}
2. $N \rightarrow CH$: {N, A_i, add, d_i, m_i}
3. $CH \rightarrow PDP$: {CH, A_i, add, d_i, m_i}
4. $PDP \rightarrow CH$: { PDP, A_i, d_i, add}
5. $CH \rightarrow PIP$: {CH, A_i, d_i, add }
6. $PIP \rightarrow CH$: {PIP, A_i, d_i, add }
7. $CH \rightarrow PDP$: {CH, A_i, d_i, add }
8. $PDP \rightarrow PAP$: { PDP, A_i, d_i, add}
9. $PAP \rightarrow PDP$: { PAP, ps_i}
10. $PDP \rightarrow CH$: { PDP, permit/ deny, obl}
11. $CH \rightarrow N$: { CH, permit/ deny, obl}
12. $N \rightarrow S_m$: { N, add, d_i, m_i}
13. $S_m \rightarrow N$: {S_m, OK}
14. $N \rightarrow PDP$: {N, (A_i, revoke, add), (Aj, grant, add)}

Actions at Each Step

1. $A_i \rightarrow N$: The MPMSD client A_i submits to the MPMSD arbiter N the comment m_i to be added to document d_i.

2. $N \rightarrow CH$: The MPMSD arbiter N then requests the Context Handler for an authorization decision.

3. $CH \rightarrow PDP$: The Context Handler sends the XACML request *add* to the PDP for *authorization decision.*

4. $PDP \rightarrow CH$: PDP sends a request to the Context Handler for collecting the information about the attributes of subject, resource and action (A_i, d_i, add).

5. $CH \rightarrow PIP$: The Context Handler then sends the request to the PIP for information about the attributes of subject, resource and action (A_i, d_i, add).

6. $PIP \rightarrow CH$: The PIP gives all the information about the attributes of subject, resource and action (A_i, d_i, add) to the Context Handler.

7. $CH \rightarrow PDP$: The Context Handler then gives the information about the attributes of subject, resource and action (A_i, d_i, add) to the PDP.

8. $PDP \rightarrow PAP$: The PDP requests for policySets ps_i to the PAP.

9. $PAP \rightarrow PDP$: The PAP then writes policySets ps_i to the PDP which will be matched with the attributes of subject, resource and action (A_i, d_i, add) for an authorization decision.

10. $PDP \rightarrow CH$: After matching the attributes with the policySets ps_i the PDP sends an authorization decision either permit or deny with some obligations to the Context Handler.

11. $CH \rightarrow N$: The Context Handler then sends the authorization decision either permit or deny with some obligations to the arbiter N.

12. $N \rightarrow S_m$: The MPMSD arbiter N then submits the comment m_i to be added to the document d_i to the storage manager S_m.

13. $S_m \rightarrow N$: The storage manager S_m then gives an OK report to the arbiter N.

14. $N \rightarrow PDP$: Finally, the arbiter N requests the PDP to revoke from A_i add comment privilege and grant the same privilege to A_j, the next reviewer already predefined in the arbiter. PDP revokes and grants accordingly.

5.8 Chapter Summary/Discussion

This chapter discusses the issue of synchronization of authorization flow with the work object flow in a workflow problem in general and DPW in particular. The synchronization is shown in this chapter in architecture level, protocol level exploiting obligation mechanism available in XACML standard. It is evident from the above discussion is that a central arbiter is needed to give the solution of synchronization of authorization flow with a work object flow. The discussion is limited to synchronization of orchestrated web services with XACML only. However, synchronization of choreographed Web services with XACML is another interesting area to be explored. This remains to be the future direction of research.

Chapter 6

Extra-Tree: A Model to Organize Execution Traces of Web Services

In this chapter, a non-linear model called Extra-Tree is proposed to organize execution traces of orchestrated Web services. This proposed model provides us a logging system which records the history of all activities from the initiation to the completion of a Web service. The main focus of this chapter is to organize execution traces of Web services in a distributed computing paradigm in the form of a tree. One of the special characteristics of this model is that the execution traces of Web services can be retrieved from the coarse-grained level to the fine-grained level of the tree as per the requirement.

6.1 Introduction

XML security standards, such as WS-security [30], SAML [67], XML signature [22] and XML encryption [68] are powerful tools for people to build security architectures in distributed systems to deliver message level authentication, integrity and encryption services. These standards offer new and robust ways to protect Web services, but they don't provide services for audit logging [56]. An interesting challenge is how to handle logging in distributed environments such as those of Web services. Web services architecture should have its own logging mechanism, so that each component involved in that particular architecture has a uniform way of creating a log for later access from a single point. The best way to perform logging in Web services architecture might be a fully distributed logging

mechanism. Both successful and failed attempt to get access to data or services must be logged. The net result of the Web services logging problem is both a challenge and an opportunity [56].

Audit logging is a vast area of research. The spectrum of types of data to be logged is wide. Depending on the perspectives or the requirements, like security, system management, event analysis etc., different types of data are to be logged. The scope of this chapter is limited to logging execution traces of orchestrated Web services. This chapter proposes a model to organize execution traces of orchestrated Web services in a tree structure, named Execution Trace Tree, *Extra-Tree* in short. In the proposed model each Web service involved in a composition has a uniform way of creating a log of execution traces for later access from a single point. The execution traces are logged in a distributed way. A special characteristic of the hierarchical model is that execution traces can be accessed in varying granularities from coarse-grain to fine-grain.

6.2 Audit Trail

An audit trail is a record showing who has accessed a computer system and what operations she has performed during a given period of time [55]. Audit trails are useful both for maintaining security and for recovering lost transactions [52], [53], [54]. Audit trail is a sequential record of computer system activities saved to a file on the system [59]. Secure logging [57] means the time of access, when it has accessed, who has accessed, what has accessed and their details are traced or logged. In non-repudiation logging [58], once a user is accessing a work object, next time it cannot be denied which has already traced. A secure audit log is also required for non-repudiation [58].

Audit trails are basically the protocol about what happened at a specific service. Audit trails are providing the following capabilities –

1) They guarantee non-repudiation, i.e. if dealing with a customer order, the manufacturer can prove that a certain order was placed by a certain principal.
2) They also give the requester a tool by which he can review (double check) his orders.
3) Audit trails should provide enough qualitative information to re-build the business transaction in case of problems like attacks, disk crashes etc.

6.3 Execution Traces

Traces can contain different information and can contain different types of information depending on what is being traced and the purpose of the trace. An execution trace can be used to describe the interacting modules involved in a particular scenario or may be detailed to capture the performed statements in each module's procedure.

The execution traces form a subset of audit trails. In execution trace only the current executable data are recorded from the time of initiation to the time of completion. The information about the system activities is typically represented using execution traces. The system activities such as login and logout, insertion, deletion, modification, date and time of action are normally traced. Execution traces represent the sequence of execution in a running software system at different levels of abstraction [65], [66]. Traces may exist in different structures according to the degree of abstraction requested by the program analyzer. Routine-level traces depict the sequence of routine calls for a program and often represented as a tree structure. The execution traces need to be captured in different levels: coarse-grain level to the fine-grain level and resources accessed during operations. This multi-level capturing of traces is useful for security issues like authorization, auditing, non-repudiation etc.

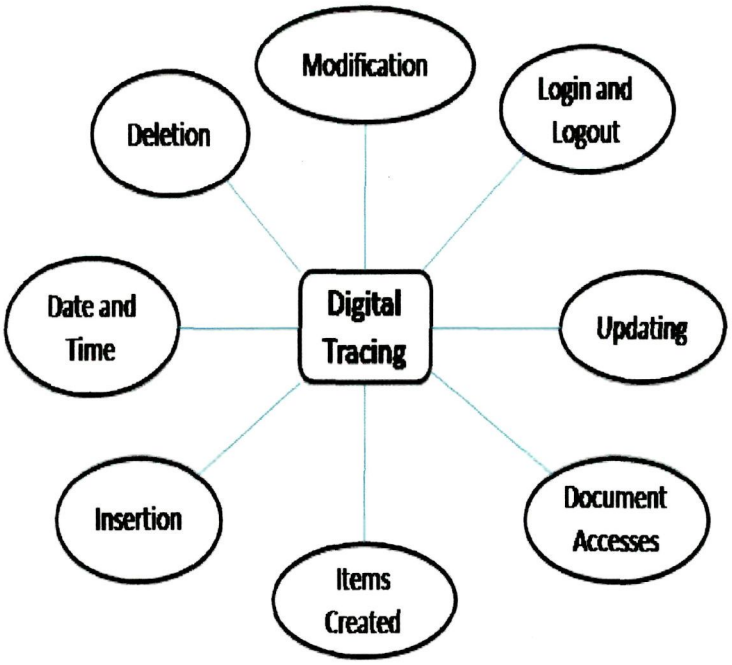


Figure 6.3: Execution Traces

6.4 BPEL Process

BPEL (Business Process Execution Language) is a language for Web services to model Web services based business processes. The core concept is the representation of peer-to-peer interactions between a process and its partners using Web services and an XML-based grammar. It is built on top of WSDL. BPEL is a language based on XML that allows control of the process flow of a set of collaborating Web services. It defines interactions that exist within and between organization processes. The language uses either a graph based or algebraic representation and offers the ability to manage both abstract and executable processes. BPEL offers an interesting feature that allows independent representation of the interactions between the partners. The interaction protocols are called abstract processes and they are specified in business protocols.

BPEL basic activities are handled by three types of messages. These are –

1) <*invoke*> : to make an operation on a partner.

2) <*receive*> : to receive an invocation from a partner.

3) <*reply*> : to send a reply message in partner invocation.

The <*assign*> activity is used to copy data from one variable to another.

BPEL is today the state of the art technology solution for Web Service orchestration. BPEL specifies a composite Web Service composed of other Web services or BPEL processes or the both. It's basically an integration artifact for integrating Web services. BPEL engine accomplishes the task as per the BPPEL specification of the business process. An important issue is to model the execution traces of BPEL process which comprises of the local activities within the BPEL process as well as remote Web services linked in partner links. The execution traces need to be captured in different levels: process, activity, operations and resources accessed during operations. This multi-level capturing of traces is useful for authorization auditing, non-repudiation, dependability analysis etc. In this chapter, it has proposed a tree model for execution traces of BPEL processes which facilitates multi-level trace analysis and which can be implemented easily both in centralised as well in distributed version using XML. More details about WS-BPEL are available in [29].

6.5 Related Works

In the paper [60], authors have given more importance of how the old information is either deleted or stored in the audit trail in a systematic manner like the current data. In addition to the actual recording of all events that takes place in the database, an audit trail must also provide query support for auditing. In this paper [60], authors used a database activity model in which relational database technique is used. The benefit of using this model is that it not only stores the current data but also gives more importance on past information which is either deleted or stored in the database in a systematic manner. Another benefit of using this model is to providing mechanism for recording and querying accesses to the database. Pau et. al proposed [53] a Data Warehouse model for capturing audit trail data and analysis. The benefits of using these models is that the managers can evaluate quality of a business processes by monitoring the performance of the operations. From this, managers and decision makers can improve existing workflow models and can optimize the performance of the business processes. In the paper [61], authors proposed a framework for secure logging in a public communication network system which are more resilient to identify security threats and also responsible to verify the integrity of the log files. In the paper [62], authors proposed several issues which have resolved for a secure logging system. These issues are related to secure logging of decentralized cross-organizational workflows and underlying decentralized data storage. Fedaghi and Mahdi [52] proposed a conceptual model that produces an implementation independent logging scheme to monitor events. In this paper [52], authors proposed a model known as Flow-Based Logging Model which provides a conceptual foundation for the classification of information related events. Helmen and Liepins [63] presented a statistical model for the detection of computer misuse using stochastic process. The benefit of using this model is to give perfect information against computer misuse detection. In the paper [64], authors presented a new approach to providing audit logs for transaction processing systems that can effectively and efficiently detect tempering.

From the above study, it is found that the existing secure logging systems mainly protect the log files from external attacks and also it can give the information about suspicious or malicious activities. But the secure logging system of Web services in the distributed computing system is not much available in the literature. So, a tree model is proposed to organize and retrieve execution traces of Web services from coarse-grain level to fine-grain level.

6.6 Extra-Tree Model

This section discusses about the proposed Extra-Tree model to organize execution traces. The scope of this model is limited to orchestrated Web services only. Elementary Web services may be implemented in any platform but can be called through its published WSDL interface. These elementary Web services may participate in many composite Web services. Thus a composite Web service is composed of one or more Elementary Web services. The composite Web services may be implemented in BPEL. Each composite Web service has also a WSDL interface through which it can be called and thereby can participate in many composite Web services. Thus a composite Web service may be composed of other composite Web services or elementary Web services. Composition and participation creates a many-to-many relationship. Web service is a distributed computing paradigm in which capturing the many-to-many relationship in a central table is against the fundamental characteristic of the paradigm. In this model, it is assumed that each Web service has two lists: a Participation List (PL) and a Composition List (CL). The PL of a service contains the unique identifiers, like URIs, of the Web services where the service in question has participated in. For a top level service the PL may be null. Similarly CL of a service contains the identifiers of the services participated in the composition of the service in question. The CL of an elementary service is null. In addition to these two lists, each service has an Execution Trace Table (ETT), which stores the execution traces of the concerned Web service. Execution trace contains the attribute values of Transaction Number, who called the Web service (another service, role or user), when called and when the call ended etc. For simplicity of the discussion, we exclude other attributes like resources used during the instance etc. But such additional audit trails can be easily added without any loss of generality.

Now, consider any top level service and track down according to the composition lists until we find elementary services, the process yields a tree. The top level service is the root, elementary services constitute the leaves and intermediate nodes are composite services.

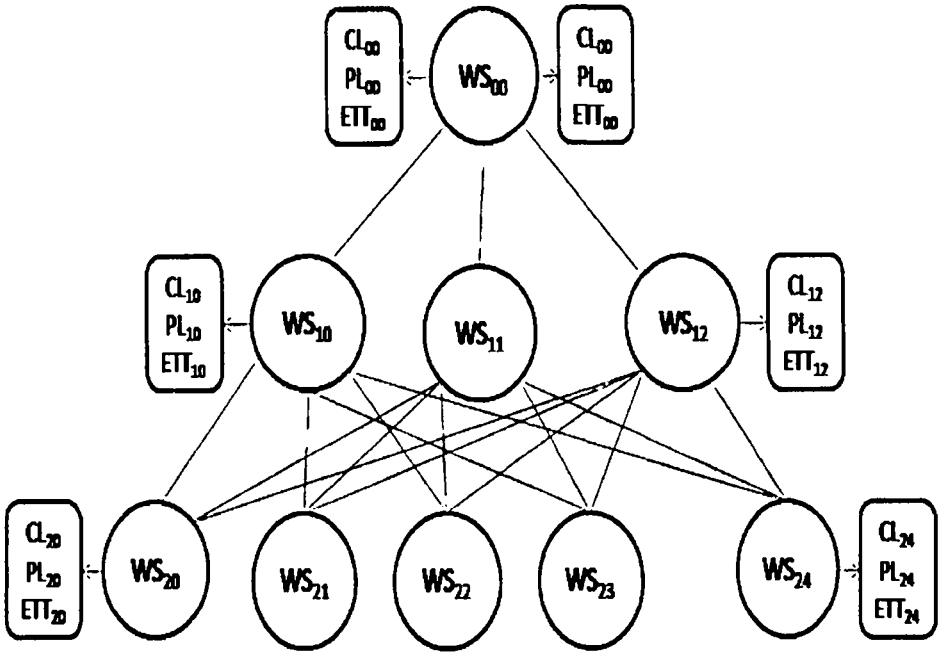


Figure 6.6 (a): The Extra Tree Model

An interesting feature of Extra-Tree model is that the execution traces of Web services are organised in the ETTs of different levels of the tree model with varying granularities. Parent node contains the coarser-grain traces than the children, or in other words, children contain finer-grain traces than the parent node. This multi-level organization enables us to retrieve the execution traces in required details only. Unless finer details are required, ETTs stored in nodes of lower depths will satisfy the query requirement. Only when complete execution history is required, the tree is traversed in full. If the knowledge is considered at best, this is the first non-linear hierarchical model to organize execution traces in Web services domain.

In Figure 6.6 (a), root Web service is WS_{00}, the execution traces in this Web services can be organized as -

$$CL_{00} = (WS_{10}, WS_{11}, WS_{12}) \text{ and } PL_{00} = NULL, ETT_{00} = \{(T, U, t_i^{00}, t_c^{00},..), (...)\},$$

Where, CL_{00}, PL_{00} and ETT_{00} are the composition list, participation list and execution trace table of the root Web service WS_{00} and T is the transaction number, U is who called and t_i^{00}, t_c^{00} ..... are the time of initiation and time of completion of the transaction.

The composite Web services in Figure 6.6 (a) are WS_{10}, WS_{11} and WS_{12}. The execution traces in this Web services can be organized as –

$CL_{10} = (WS_{20}, WS_{21}, WS_{22})$, $PL_{10} = (WS_{00}, WS_{01}, WS_{02}$ ----) and $ETT_{10} = \{(T, WS_{00}, t_i^{10}, t_c^{10} \ldots.), (\ldots)\}$,

$CL_{11} = NULL$, $PL_{11} = (WS_{00}, WS_{01}, WS_{02}$ ----) and $ETT_{11} = \{(T, WS_{00}, t_i^{11}, t_c^{11} \ldots.), (\ldots)\}$,

$CL_{12} = (WS20, WS_{21}, WS_{22})$, $PL_{12} = (WS_{00}, WS_{01}, WS_{02}$ ----) and $ETT_{12} = \{(T, WS_{00}, t_i^{12}, t_c^{12} \ldots.), (\ldots)\}$,

Where, CL_{10}.....CL_{12}, PL_{10}.....PL_{12} and ETT_{10}.....ETT_{12} are the composition list, participation list and the execution trace table of the composite Web services WS_{10}, WS_{11} and WS_{12}. Here T is the transaction no., WS_{00} is who called and t_i^{10}, t_c^{10}.....t_i^{12}, t_c^{12} are the time of initiation and time of completion of the transaction.

The elementary Web services in Figure 6.6 (a) are WS_{20}, WS_{21}, WS_{22}, WS_{23} and WS_{24}. The execution traces in this Web services can be organized as –

$CL_{20} = NULL$, $PL_{20} = (WS_{10})$ and $ETT_{20} = \{(T, WS_{10}, t_i^{20}, t_c^{20}\ldots.), (\ldots)\}$,

$CL_{21} = NULL$, $PL_{21} = (WS_{10})$ and $ETT_{21} = \{(T, WS_{10}, t_i^{21}, t_c^{21}\ldots.), (\ldots)\}$,

$CL_{22} = NULL$, $PL_{22} = (WS_{10})$ and $ETT_{22} = \{(T, WS_{10}, t_i^{22}, t_c^{22}\ldots.), (\ldots)\}$,

$CL_{23} = NULL$, $PL_{23} = (WS_{12})$ and $ETT_{23} = \{(T, WS_{12}, t_i^{23}, t_c^{23}\ldots.), (\ldots)\}$,

$CL_{24} = NULL$, $PL_{24} = (WS_{12})$ and $ETT_{24} = \{(T, WS_{12}, t_i^{24}, t_c^{24}\ldots.), (\ldots)\}$,

Where, CL_{20}.....CL_{24}, PL_{20}.....PL_{24} and ETT_{20}.....ETT_{24} are the composition list, participation list and the execution trace table of the elementary Web services WS_{20}, WS_{21}, WS_{22}, WS_{23} and WS_{24}. Here, T is the transaction no. WS_{10} and WS_{12} are who called and t_i^{20}, t_c^{20}.....t_i^{24}, t_c^{24} are the time of initiation and time of completion of the transaction.

For a given transaction T, the PLs, CLs and ETTs are tabulated as follows –

WS_{00}

PL_{00}

NULL

CL_{00}

WS_{10}	WS_{11}	WS_{12}	-	-

ETT_{00}

Who Called	**Transaction Number**	**When Called**	**When Ended**	-
U	T	t_i^{00}	t_c^{00}	-

Table 6.6 (a): Execution Traces of Web service WS_{00}

WS_{10}

PL_{10}

WS_{00}	WS_{01}	WS_{02}	-	-

CL_{10}

WS_{20}	WS_{21}	WS_{22}	-	-

ETT_{10}

Who Called	**Transaction Number**	**When Called**	**When Ended**	-
WS_{00}	T	t_i^{10}	t_c^{10}	-

Table 6.6 (b): Execution Traces of Web service WS_{10}

WS_{11}

PL_{11}

WS_{00}	WS_{01}	WS_{02}	-	-

CL_{11}

NULL

ETT_{11}

Who Called	**Transaction Number**	**When Called**	**When Ended**	-
WS_{00}	T	t_i^{11}	t_c^{11}	-

Table 6.6 (c): Execution Traces of Web service WS_{11}

WS_{12}

PL_{12}

WS_{00}	WS_{01}	WS_{02}	-	-

CL_{12}

WS_{23}	WS_{24}	-	-	-

ETT_{12}

Who Called	**Transaction Number**	**When Called**	**When Ended**	-
WS_{00}	T	t_i^{12}	t_c^{12}	-

Table 6.6 (d): Execution Traces of Web service WS_{12}

WS_{20}

PL_{20}

WS_{10}	-	-	-	-

CL_{20}

NULL

ETT_{20}

Who Called	**Transaction Number**	**When Called**	**When Ended**	-
WS_{10}	T	t_i^{20}	t_c^{20}	-

Table 6.6 (e): Execution Traces of Web service WS_{20}

WS_{21}

PL_{21}

WS_{10}	-	-	-	-

CL_{21}

NULL

ETT_{21}

Who Called	**Transaction Number**	**When Called**	**When Ended**	-
WS_{10}	T	t_i^{21}	t_c^{21}	-

Table 6.6 (f): Execution Traces of Web service WS_{21}

WS_{22}

PL_{22}

WS_{10}	-	-	-	-

CL_{22}

NULL

ETT_{22}

Who Called	**Transaction Number**	**When Called**	**When Ended**	-
WS_{10}	T	t_i^{22}	t_c^{22}	-

Table 6.6 (g): Execution Traces of Web service WS_{22}

WS_{23}

PL_{23}

WS_{12}	-	-	-	-

CL_{23}

NULL

ETT_{23}

Who Called	**Transaction Number**	**When Called**	**When Ended**	-
WS_{12}	T	t_i^{23}	t_c^{23}	-

Table 6.6 (h): Execution Traces of Web service WS_{23}

WS_{24}

PL_{24}

WS_{12}	-	-	-	-

CL_{24}

NULL

ETT_{24}

Who Called	**Transaction Number**	**When Called**	**When Ended**	-
WS_{12}	T	t_i^{24}	t_c^{24}	-

Table 6.6 (i): Execution Traces of Web service WS_{24}

For a given transaction, symbolically the parenthesis structure can be written as –

$$\begin{matrix} When\ called \\ Who\ called \end{matrix} \left(\quad \right) \begin{matrix} When\ ended \\ Who\ called \end{matrix}$$

Figure 6.6 (b): The Parenthesis Structure

Therefore, the full execution of a transaction in the form of Well-formed expression can be written as –

$$t_i^{00}\left(t_i^{10}\left(t_i^{20}\left(\right)_{ws_{20}}^{t_c^{20}} \; t_i^{21}\left(\right)_{ws_{21}}^{t_c^{21}} \; t_i^{22}\left(\right)_{ws_{22}}^{t_c^{22}} \right)_{ws_{10}}^{t_c^{10}} \; t_i^{11}\left((-) \right)_{ws_{11}}^{t_c^{11}} \; t_i^{12}\left(t_i^{23}\left(\right)_{ws_{23}}^{t_c^{23}} \; t_i^{24}\left(\right)_{ws_{24}}^{t_c^{24}} \right)_{ws_{12}}^{t_c^{12}} \right)_{ws_{00}}^{t_c^{00}}$$

Figure 6.6 (c): The Well-formed Expression

In the above expression, WS_{00}, WS_{10} .....WS_{24} are the root web services, composite Web services and elementary Web services respectively and t_i^{00}, t_c^{00}..... are the required time interval in each of the level of Web services to organise and retrieve them from the coarse-grained level to the fine-grained level of the tree model. The motivation behind this expression is that it can be easily implemented in nested XML elements with attributes. Now, to generate this expression we have to consider Depth-First-Search technique.

Execution traces of Web services are retrieved from Extra-Tree by RetrieveTrace algorithm. It is essentially a modified Depth-First-Traversal (DFT) algorithm. For coarse-grain traces there is no need to traverse the full tree. Therefore, the depth to be traversed is to be given as input to the algorithm. Depth 0 means root only, any number greater than 0 is the absolute maximum depth. Actual depth of the tree is less than the input maximum depth implies the full tree traversal. The RetrieveTrace algorithm is given below.

Algorithm RetrieveTrace(W, T, L) //algorithm to retrieve execution traces

1. T: Transaction No, L: Depth //declaring global variables
2. l := 0; // initializing l
3. DFT_VISIT(W) // to start visit the root Web service W

Algorithm: DFT_VISIT(W: WS) // algorithm for Depth-First-Traversal

1. {S: WS;
2. mark[W] := *visited* // root Web service W is already visited
3. Trace := select (ETT(W), T); // who called, transaction no. etc. will be selected from ETT
4. Print W/C(t_i; // print the traces of service who called, time of initiation
5. Push(stack,)t_c); // push traces into the stack
6. l := l+1 // incrementing l
7. **While** (l < = L) **do**
8. { For each S in PL[W] from left to right order
9. If mark[S] = *unvisited* then
10. DFT_VISIT(S);
11. }

12. T_c := pop((stack,)t_c); // pop traces from the stack and stores in a temporary variable

13. Print T_c; // Finally, the Well-formed parenthesis is printed

14. }

Table 6.6 (j): Algorithm RetrieveTrace

6.7 Chapter Summary/Discussion

This chapter discusses an Extra-Tree Model which organizes and retrieves execution traces of Web services in different levels. The retrieval of execution traces of Web services is shown by an algorithm using Depth-First-Search technique. The advantage of this model is to retrieve the execution traces from the coarse-grained level to the fine-grained level in different time intervals. Normally, all execution traces are linear but in this model traces are non-linear.

From this chapter, it is assumed that the ETTs, PLs and CLs are securely stored in the concerned Web services and resilient to attacks. But from the security point of view it is a costly assumption. If audit data stored in a distributed way are compromised then the whole objective of audit analysis will be meaningless. Therefore, the organization of execution traces of Web services in a central point rather than in a distributed way will probably give the more secured auditing system and make the proposed model robust which will be more challenging as a future research work.

Chapter 7

Conclusions

This chapter concludes the thesis by summarizing the findings and the contributions of this thesis and presents several directions for future research.

7.1. Summary of Works

The security requirements are the types and levels of protection necessary for equipment, data, information, applications and facilities to meet a security policy. A secure Web service is defined as a computer supported business process that satisfies the security requirements in a workflow environment. In this thesis, the following aspects of security issues in Web services are addressed.

Chapter 2 discusses on the overview of Web service and security. The chapter mainly reviews the security aspects of Web services. It is seen that the security concerns are similar to other web applications in general; however Web services have specific security issues as well. The specific security challenges are mainly due to the adoption of XML in different components of Web services. The present study includes recent threats and attacks specific to XML-based Web services.

Chapter 3 discusses on the limitations of Web services security during SOAP message transmission. The main focus of this chapter is to show the limitations of WSS on the SOAP message integrity and also to show how the issue can be addressed by a protocol. From the discussion of this chapter it is clear that it is not possible to address the integrity issue, as

claimed in the standard [30], [31] within the scope of message level. A central arbitration mechanism is mandatory. In case of orchestrated Web service based workflow, BPEL process can take the responsibility of this arbitration along with coordination of constituent Web services. However, in case of a complex workflow across many business entities, where multiple BPEL processes are to be co-ordinated, different protocol may have to be used in addition.

Chapter 4 discusses about the SWOT analysis of 2-tuple digital signature. In public-key based 2-tuple digital signature, the signature and the message content are loosely bound. In one hand this signature scheme is adventitious from efficiency point of view but on the other hand it invites vulnerability from security point of view. Present digital signature standard is not resilient to signature replacement attack. This chapter proposes a solution in the protocol level with a central arbiter as an in-line Trusted Third Party (TTP) mechanism. . The central arbitration mechanism is mandatory to address such issues.

Chapter 5 presents the synchronization of authorization flow with a work object flow. The synchronization is shown in this chapter in architecture level and protocol level exploiting obligation mechanism available in XACML standard. The chapter shows that a central arbiter is needed to give the solution of synchronization of authorization flow with a work object flow. The discussion is limited to synchronization of orchestrated web services with XACML only.

Chapter 6 proposes a model called Extra-Tree for organising execution traces of Web services. The advantage of this model is to retrieve the execution traces of Web services from the coarse-grained level to the fine-grained level at different time intervals. This is the first non-linear hierarchical model to organize execution traces in Web services domain, hitherto not discussed in literature.

7.2. Summary of Contributions

The main contributions of this thesis include the development of some proposed solutions, architecture and protocols, model with algorithm in connection with different security issues.

It is found from the investigation that there are some more limitations of current WSS standard on SOAP messages. Current version of WSS standard fails to address these issues.

In WSS standard, XML signatures and XML encryption with inclusive and exclusive canonicalization ensure proof of origin, content integrity and confidentiality issues of each individual part, that is, each signed message, in a SOAP envelope during exchange and storage. But it fails to ensure the collective integrity issue of all the parts as a whole, which means all the signed messages in a SOAP envelope signed by the original sender and the intermediaries during the workflow. It also fails to address the part reuse issue. WSS standard fails to ensure end-to-end security in SOAP message level on these issues. However, a solution with a special protocol exists. The protocol is based on a central arbitration mechanism. It is basically a client/server computing paradigm, where the arbiter is the server. To address these issues an in-line Trusted Third Party (TTP), called an arbiter, is mandatory. The arbiter serves as the trusted intermediate agent in between the current reviewer and the next.

The weaknesses of 2-tuple digital signature scheme pave the way for signature replacement attack, which is not discussed in the literature. It seems that no solution is possible keeping the loose binding of the message and the signature. A counter-measure is to create a digital envelope encapsulating the signature and the message. But the use of digital envelope also fails to give the solution of this attack which is secured during transmission not in storage. However, a solution with a special protocol exists. This protocol is based on a central arbitration mechanism. All the messages flowing from the originator to the next receiver are routed through the arbiter. To address these issues an in-line Trusted Third Party (TTP), called an arbiter is mandatory. The arbiter serves as the trusted intermediate agent in between the originator and the receiver.

The solution of synchronization of authorization flow with a work object flow has shown in two levels – the architecture level and the protocol level. In the conceptual architecture level, a 3-tier architecture is proposed. In the protocol level, the merging of the protocol for work object flow and the protocol for authorization flow are discussed.

Finally, an Extra-Tree model is proposed to organize execution traces of Web services. In this chapter an algorithm is proposed to retrieve execution traces in different levels of granularities. The scope of this model is limited to orchestrated Web services only.

7.3. Future Research

In this thesis, the study and analysis of security issues in service-oriented computing are done particularly on authorization of Web services, integrity of Web services, 2-tuple digital signature replacement attack and the model for execution traces of Web services.

For the limitations of WSS a solution is proposed by using an in-line TTP. The future research study will be in the direction of choreographed workflow implementations. The implementation of the protocol in both orchestrated and choreographed workflows in Web service world is the direction of future research. For the signature replacement attack issue, a solution is proposed by the use a central arbiter mechanism. The future works include the detail study on signature replacement attack and its counter-measures in both orchestrated as well as choreographed Web services. The synchronization of authorization flow with a work object flow using XACML and BPEL is shown in architecture level and protocol level. The coding level synchronization of both XACML and BPEL is the future research work. The synchronization of choreographed Web services with XACML is also another interesting area to be explored, which remains to be the future research work. In the Extra-Tree model, the organization of execution traces of Web services in a central point rather than in a distributed way will probably give the more secured auditing system and make the proposed model robust which is a future endeavour.

Bibliography

[1] Denning, D. E.; Bargh, W. E. (1999). *Easy Guide to Encryption.* Expert Control, Sept. 25.

[2] Huhns, M. N.; Singh, M. P. (2005). *Service Oriented Computing: Key Concepts and Principles.* IEEE Internet Computing, pp. 75-81.

[3] Srivastava, B.; Koehler, J. (2004). *Web Service Composition: Solutions and Open Problems.* IBM India Research Laboratory.

[4] Milanovic, N.; Malek, M. (2004). *Current Solutions for Web Service Composition.* IEEE Internet Computing.

[5] Forman, G. H.; Zahorjan, J. (1994). *The Challenges of Mobile Computing.* IEEE.

[6] Yang, H.; Luo, H.; Ye, F.; Zhang, L. (2004). *Security in Mobile Ad-hoc Networks: Challenges and Solutions.* IEEE Wireless Communication.

[7] Yu, Q.; Liu, X.; Bouguettaya, A.; Medjahed, B.(2006). *Deploying and Managing Web Services: Issues, Solutions and Directions.* Springer Verlag, pp. 537-572.

[8] Barbir, A.; Hobbs, C.; Bertino, E.; Hirsch, F.; Martino, L.: Challenges of Testing Web Services and Security in SOA Implementations. Springer Verlag, Book Chapter, pp. 395-440.

[9] Wang, L.; Lee, L. *Modelling and Construction of Web Services Security.* Springer Verlag, Book Chapter, pp. 273-282.

[10] Benatallah, B.; Perrin, O.; Rabhi, F. A.; Godart, C. *Web Service Computing: Overview and Directions.* Springer Verlag, Book Chapter, pp. 553-574.

[11] Pimenidis, E.; Georgiadis, C. K.; Bako, P.; Zorkadis, V. (2008). *Web Services Security: Implementation and Evaluation Issues*. Springer Verlag, pp. 299-308.

[12] Simple Object Access Protocol 1.1. (2000), www.w3c.org/TR/soap/....

[13] Benameur, A.; Kadir, A. F.; Fenet, S. (2008). *XML Rewriting Attacks: Existing Solutions and their Limitations.* In: IADIS Applied Computing. IADIS press.

[14] Hermann, E.; Kessler, D. (2005): *XML Signatures in an Enterprise Service Bus Environment.* International Federation for Information Processing. LNCS, pp. 339-347.

[15] Neil, O' M. (2003). *Web Services Security.* TATA Mcgraw Hill Pub. Newyork pp. 04-22.

[16] Singhal, A.; Winograd, T.; Scarfone, K. (2007). *Guide to Secure Web Services.* NIST.

[17] Teraguchi, M.; Makino, S.; Ueno, K.; Chung, H. V. (2006): *Optimized Web Services Security Performance with Differential Parsing.* Springer Verlag, pp. 277-288.

[18] Chevalier, Y.; Lugiez, D.; Rusinowitch, M. (2007). *Towards an Automatic Analysis of Web Service Security.* Springer Verlag, pp. 133-147.

[19] oasis-xacml1.0, (2003). www.oasis-open.org/committees/xacml/repository/

[20] Jensen, M.; Guruschka, N.; Herkenhoner, R. (2009). *A Survey of Attacks on Web Services: Classification and Countermeasures.* Springer Verlag, pp. 185-197.

[21] Kearney, P.; Chapman, J.; Edwards, N.; Gifford, M.; He, I. (2004). *An Overview of Web Services Security.* BT Tech. Journal, Vol. 22, No. 1.

[22] XML Signature Syntax and Processing, (2008). www.w3.org/TR/xmldsig-core/

[23] Chadwick, D. W.; Otenko, S.; Nguyen, T. A. (2006). *Adding Support to XACML for Dynamic Delegation of Authority of Multiple Domains.* IFIP, LNCS, pp. 67-86.

[24] Web Service Description Language (2001). [Online]. Available: http://www.w3c.com/TR/wsdl/.

[25] Universal Description Discovery Integration. www.w3c.com/uddi/....

[26] Sinha, S. K.; Benameur, A. (2008). *A Formal Solutions to Re-writing Attacks on SOAP Messages*. ACM International Workshop on Secure Web Services. ACM, Virginia USA pp. 53-59.

[27] Moradian, E.; Hakansoon, A. (2006). *Possible Attacks on XML Web services*. International Journal of Computer Science and Network Security, Vol. 6. pp. 154-170.

[28] Mercuri, R. T. (2003). *Security Watch on Auditing Audit Trails*. ACM, Vol. 46, No. 1, pp. 17-20.

[29] Business Process Execution Language v2.0 (2007). [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

[30] WS-Security (2004), OASIS standard (2004). [Online]. Available: http://docs.oasisopen.org/wss/v1.1/

[31] Web Services Security (2006). [Online]. Available: http://www.oasisopen.org/committees/wss/

[32] Khoury, P. E. (2008). *XACML as Security and Integration Patterns*. In: Proc. of 2nd International Conference on Ambient Intelligence Development. Springer Verlag, pp. 143–155.

[33] McIntosh, M.; Austel, P. (2005). *XML Signature Element Wrapping Attacks and Counter-measures*. In: Proceedings of the 2005 Workshop on Secure Web services. New York, USA: ACM, pp. 20–27.

[34] Gajek, L. L. S.; Schwenk, J. (2007). *Breaking and Fixing the Inline Approach*. In: Proceedings of the 2007 ACM Workshop on Secure Web Services. New York, USA: ACM, pp. 37–43.

[35] Rahaman, M. A.; Schaad, A.; Rits, M. (2006). *Towards Secure SOAP Message Exchange in a SOA*. In: Proceedings of the 3rd ACM workshop on Secure Web Services. New York, USA: ACM, pp. 77–84.

[36] Sinha, S. K.; Barua, G. (1999). *An Architecture for Document Production Workflow in an Office*. In: Proc. of International Conference on Information Technology (CIT99). India, Tata McGraw Hill Publishers, pp. 45–52.

[37] Sinha, S. K.; Barua, G. (2000). *A Protocol for Secure Flow of Persistent Multi-part Documents in an Office*. In: Proc. of the International Forum cum Conference on Information Technology and Communication at the Dawn of the New Millennium. Tata McGraw Hill Publishers, India, pp. 157–172.

[38] Sinha, S. K. (2001). *Secure Production and Storage of Digital Documents in an Office Environment*. In: Ph. D. Thesis, Tezpur University.

[39] Zhou, J.; Gollmann, D. (1996). *Observations on Non-Repudiation*. In: Proc. of ASIASIACRYPT' 96. LNCS, Springer Verlag, pp. 133–144.

[40] Zhang, Z.; Feng, D. *Key Replacement Attack on a Certificateless Signature Scheme*. Information Security Institute of s/w. Academy of Sciences, Beijing, China.

[41] Sinha, S. K.; Sinha, S. (2008). *Limitations of Web Service Security on SOAP messages in a Document Production Workflow Environment*. In: Proceedings of the ADCOM'08, IEEE Xplore, pp. 242-246.

[42] Gupta, K. N.; Agarwala, K. N.; Agarwala, P. A. (2005). *Digital Signature: Network Security Practices*. PHI Publication. New Delhi.

[43] Ahmed, S.; Armstrong, L.: *Securing Web Services with XML Aware Digital Signature*. School of Computer and Information Sc., Edith Cowan University.

[44] SWOT Analysis, www.netmba.com/strategy/SWOT/.

[45] Relational Database System, (2001). www.gradkell.com/pdf/DBsecurity/article

[46] Java Security, www.docstore.mik.ua/ovelly/java-ent/security/ch1201.htm.

[47] Dhankhar, V.; Kaushik, S.; Wijesekera, D. (2008). *Securing Workflows with XACML, RDF and BPEL*. International Federation for Information Processing (IFIP), LNCS, pp. 330-345.

[48] Wang, X.; Zhang, Y.; Shi, H.; Yang, J. (2008). *BPEL4RBAC: An Authorization Specification for WS-BPEL*. Springer-Verlag Berlin Heidelberg, LNCS, pp. 381-395.

[49] Paci, F.; Bertino, E.; Crampton, J. (2008). *An Access-Control Framework for WS-BPEL*. International Journal of Web Service Research, Vol. 5, Issue 4. pp. 20-43.

[50] Sanchez, M.; Lopez, G.; Gomez, A. F.; Canovas, O. (2006). *Using Microsoft Office Infopath to Generate XACML Policies*. Springer-Verlag Berlin Heidelberg, ICETE 2006, CCIS 9, pp. 134-145.

[51] Juric, M. B.; Mathew, B.; Sarang, P. (2006). *Business Process Execution Language for Web Services*. Packt Publication Ltd., Birmingham, UK. ISBN 1-904811-81-7.

[52] AI-Fedaghi, S.; Mahdi, F. (2010). *Events Classification in Log Audit*. International Journal of Network Security and its Applications, (IJNSA), Vol. 2, No. 2.

[53] Pau, K. C.; Si, Y. W.; Dumas, M.: *Data Warehouse Model for Audit Trail Analysis in Workflows*. http://citeseerx.ist.psu.edu/viewdoc/.....

[54] Duong, M. (2009). *Audit Trail for SAS Data Sets*. Texas Institute of Measurement, Evaluation and Statistics. University of Houston. www.scsug.org/scsug Proceedings/2009/Minh_Duong.pdf/.....

[55] Onwubiko, C. (2009). *A Security Audit Framework for Security Management in the Enterprise*. Springer-Verlag Berlin Heidelberg, pp. 9-17.

[56] Steven, J.; Peterson, G.; Frincke, D. A. (2009). *Logging in the age of Web services*. IEEE Security and Privacy Journal, pp. 82-85.

[57] Kent, K.; Souppaya, M. (2006). *Guide to Computer Security Log Management*. Special Publication. Computer Security Division, IT Lab. Gaithenburg.

[58] Tsai, C. R.: *Non–Repudiation in Practice*. Citigroup Information Security Office, Silver Spring, USA. Available in http://citeseerx.ist.psu.edu/viewdoc/.....

[59] Mercuri, R. T. (2003). *Security Watch on Auditing Audit Trails*. ACM, Vol. 46, No. 1, pp. 17-20.

[60] Jajodia, S.; Gadia, S. K.; Bhargava, G.: *Logical Design and Audit Information in Relational Databases*. Information Security, pp. 585-595.

[61] Stanthopoulos, V.; Kotzarikolon, P.; Magkos, E. (2006). *A Framework for Secure and Verifiable Logging in Public Communication Networks*. Springer, LNCS 4347, pp. 273-284.

[62] Wombacher, A.; Wieringa, R.; Jonker, W.; Knezevic, P.; Pokraev, S. (2005). *Requirements for Secure Logging of Decentralized Cross-organizational Workflow Executions*. Springer, LNCS 3762, pp. 526-536.

[63] Helman, P.; Liepins, G. (1993). *Statistical Foundations of Audit Trail Analysis for the detection of Computer Misuse*. IEEE transactions on Software Engineering, vol. 19, No. 9.

[64] Snodgrass, R. T. (2004). *Tamper Detection in Audit Logs*. In: Proceedings of the 30th VLDB Conference, Toronto, Canada, pp. 504-515.

[65] Alawneh, L.; Hamon, A. (2009). *Execution Traces: A New Domain that Requires the Creation of a Standard Metamodel*. ASEA 2009, CCIS 59, Springar, pp. 253-263.

[66] Chesani, F.; Mello, P.; Montali, M.; Riguzzi, F.; Sebastianis, M.; Stovari, S. (2009). *Checking Compliance of Execution Traces to Business Rules*. BPM 2008 Workshops, LNBIP 17, Springer, pp. 134-145.

[67] Security Assertion Markup language, (2008). www.oasis-open.org/saml/.....

[68] XML Encryption Syntax and Processing, (2002). www.w3c.org/TR/xmlencry-core/.....

[69] Hafner, M.; Breu, R. (2005). Realizing Model Driven Security for Inter-Organisational Workflows with WS-CDL and UML 2.0. LNCS, Springer, pp. 39-53.

Appendix A

Definitions of Terms

This appendix lists the terms frequently used in this thesis.

Service Oriented Computing - Service Oriented Computing (SOC) is the new emerging paradigm for distributed computing and e-business processing that is changing the way software applications are designed, architected, delivered and consumed.

Service Oriented Architecture – Service Oriented Architecture (SOA) consists of three kinds of participants: a service provider, a service requester and a service registry. The interactions involve among these participants are publish, find and bind operations.

Web Service - A Web service is a software system identified by a URL, whose public interfaces and bindings are defined and described using XML.

XML – The XML (eXtensible Markup Language) is a W3C standard markup language. It is mainly used for the implementations of Web service technologies. WSDL, BPEL, SOAP are all derivatives of XML. Therefore the foundation of Web service technology is on XML.

SOAP - Simple Object Access Protocol (SOAP) is a standard message structure used for communication among different Web Services. SOAP messages flow from originator to an ultimate receiver through a SOAP message path. It is a W3C standard available in *www.w3c.org/TR/soap*

WSDL - Web Service Description Language (WSDL) is an XML based language for describing functional properties of Web services. It is a W3C standard available in *www.w3c.org/TR/wsdl*

UDDI - Universal Description Discovery and Integration (UDDI) is an XML-based registry for Web services. It is a place where businesses register and search for Web services. It is a W3C standard available in *www.w3c.org/uddi/.....* .

BPEL - Business Process Execution Language (BPEL) is an XML-based language to specify business processes that orchestrate the operations of several Web services. It is a OASIS standard available in *www.docs.oasis-open.org/wsbpel/2.0/.....*

Workflow – A Workflow is a set of co-ordinated tasks. A Workflow Management System (WFMS) is a software that supports the specification, execution and co-ordination of tasks in a workflow.

DPW - The process of document production in an office which is based on a request-reaction-response paradigm is known as Document Production Workflow (DPW).

MPMSD - A Multi-Part Multi-Signature Document (MPMSD), D_W, produced in a DPW W, is an n-tuple, $n \geq 1$, such that $D_W = (d_1, d_2, d_3, \cdot\ \cdot\ \cdot, d_n)$. Each part d_i in turn is defined as a 3-tuple (m_i, σ_i, s_i), where m_i is the comment of the reviewer s_i, and σ_i is the signature of s_i.

Security - Security is the degree of protection against threats, attacks and vulnerability.

Threats - The possibility of an attack is known as threat.

Attacks - Any action targeting at the violation of security properties is called an attack.

Digital Signature – Digital signatures are analogs of handwritten signatures generally generated by using public-key crypto system. A digital message or its digest is encrypted with a secret key known only to the user. It can be decrypted by its conjugate public-key by

other users for verification. Digital signatures address the issues of user authentication, content integrity, non-repudiation and certification.

TTP - An inline Trusted Third Party (TTP) is an arbiter which serves as the trusted intermediate agent in between two communicating agents.

Audit Trail - An audit trail is a record showing who has accessed a computer system, resources and what operations are performed by the user concerned during a given period of time.

Execution Trace – Execution traces form a subset of audit trails. In execution trace only the current executable data are recorded from the time of initiation to the time of completion. The execution traces need to be captured in different levels like coarse-grain level to the fine-grain level and resources accessed during operations.

Extra-Tree - A model to organize execution traces of orchestrated Web services in a tree like structure, named Execution Trace Tree, Extra-Tree in short.

Obligation – An operation specified in a policy or policySet that should be performed in conjunction with the enforcement of an authorization decision.

WS-Security - WS-Security is a security framework for Web services. It is a OASIS standard for Web Services Security (WSS). It includes SOAP message security, WS-Security policy, WS-Trust, WS-Privacy, WS-SecureConversation, WS-Federation and WS-Authorization. Available in *www.docs.oasisopen.org/wss/v1.1/*.....

XML Security – XML security deals with security of XML documents, which are semi-structured and platform-independent. XML security includes XML encryption, XML digital signature, XML XML cannonicalization etc.

XML Encryption - The XML encryption enables encryption of an entire XML document or specific parts of an XML document. It is a W3C standard available in *www.w3c.org/TR/xmlencry-core/*.....

XML Signature – XML signature provides a useful means of expressing a digital signature over XML data. It is a W3C standard available in *www.w3c.org/TR/xmldsig-core/.....*

XACML - The eXtensible Access Control Markup Language (XACML) is an XML based language which is required to make authorization decisions. The decision may be permit, deny, indeterminate or error. It is a OASIS standard available in *www.oasis-open.org/committees/xacml/repository/.....*

Secure Socket Layer - SSL (Secure Socket Layer) is a technology widely used in browsers and Web servers to create a secure channel between two communicating TCP points. A common protective measure is to send messages over a secure connection using SSL.

Cross Site Scripting (XSS or CSS) - This is a type of attack where the attacker inserts malicious code in the request and this will be returned to the victim by that application.

SWOT - SWOT analysis is a strategic planning method used to evaluate the Strengths, Weaknesses, Opportunities and Threat (SWOT) involved in a particular system.

Appendix B

Symbols and Notation

This appendix lists the symbols frequently used in this thesis.

$\|\|$	Concatenation
$=$	Equality
$\leftarrow$	Assign
$\rightarrow$	communicates to
CL	Composite list
PL	Participatory list
ETT	Execution Trace Table
WS_{00}	Root Web service
WS_{10}	Child Web service
WS_{11}	Child Web service
WS_{12}	Child Web service
WS_{20}	Elementary Web service

WS_{21}	Elementary Web service
WS_{22}	Elementary Web service
t_i	Time of initiation
t_c	Time of completion
T	Transaction number
U	User/who called
m_A	An application submitted by the originator A
m_B	Comment given by the reviewer B
m_C	Comment given by the reviewer C
m_D	Comment given by the reviewer D
A_i	i^{th} Reviewer
A_{i+i}	Next reviewer
I_d	User-id
N	Arbiter
m_i	Comment of the reviewer
m_{i+1}	Comment of the next reviewer
d_i	Document
ps_i	Policy set
s_{attr}	Attributes of Subject
r_{attr}	Attributes of Resource
a_{attr}	Attributes of action

a_{dec}	Authorization decision
a_{req}	Access request
$xacml_{req}$	XACML request
PAP	Policy Access Point
PIP	Policy Information Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
CH	Context Handler
S_m	Storage Manager
obl	Obligations
SK(A)	Secret key of A
PK(A)	Public key of A
$\{\}_K$	Encryption/decryption by key
K	Key
$\{m\}_{SK(A)}$	Digital Signature
$\{\{m\}_{SK(A)}\}_{PK(A)}$	Verification of digital signature
m	Message
δ_m	Digest of a message m (encrypted)
h( )	One way hash function
h(m)	Hash function of a message

$Sk(B)$	Secret key of B
$Pk(B)$	Public key of B
$\delta/_m$	Digest of a message (decrypted)
D_W	SOAP message/document
$(d_1, d_2, d_3, \cdot \cdot \cdot, d_n)$	Previous signed messages
s_i	Reviewer given the comment m_i
σ_i	Signature of the reviewer

Appendix C

List of Publications

The following list of papers have already been published and accepted:

A. Conference

1. Subrata Sinha, Smriti K Sinha and Bipul Syam Purkayastha. "*Current Trends in Web Services and Security.*" In: Proceedings of the First National Conference on Current Trends in Computer Science, CTCS 2010, **Narosa Publishing House**, 2010. (in press)

2. Smriti K. Sinha and Subrata Sinha. "*Limitations of Web Service Security on SOAP Messages in a Document Production Workflow Environment.*" In: Proceedings of the International Conference on Advance Computing and Communications, ADCOM 2008, **IEEE Computer Society Digital Library**, pp. 232-236, 2008. (Published)

3. Subrata Sinha and Smriti K Sinha. " *Signature Replacement Attack and Its Counter-Measures*". In: Proceedings of the 2nd International Advance Computing Conference, IACC 2010, **IEEE Computer Society Digital Library,** pp. 229-235, 2010. (Published)

4. Subrata Sinha, Smriti K Sinha and Bipul Syam Purkayastha. " *Synchronization of Authorization Flow with Work Object Flow in a Document Production Workflow Using XACML and BPEL*". In: Proceedings of the International Conference on Information and Communication Technology, ICT 2010, CCIS 101, **Springer Digital Library**, pp. 365-370, 2010. (Published)

5. Subrata Sinha, Smriti K Sinha and Bipul Syam Purkayastha." *Extra-Tree: A Model to Organize Execution Traces of Web Services*". Accepted in the 6th International Conference on Next Generation in Web Service Practices, NWeSP 2010, proceedings will be published in **IEEE Digital Library**, 2010. (Accepted)

B. Journal

1. Subrata Sinha, Smriti K Sinha and Bipul Syam Purkayastha. " *Security Issues in Web Services: A Review and Development Approach of Research Agenda*". An International Journal of Science and Technology, Assam University, vol. 5. No. 11, ISSN 0975-2773, pp. 134-140, 2010. (Published)